Ein Beitrag zur digitalen Simulation kontinuierlicher Systeme auf einem Kleinrechner im Hinblick auf die Anwendung für Stromrichterschaltungen

Dissertation
zur Erlangung des Grades eines
Doktor-Ingenieurs
der Abteilung für Elektrotechnik
an der Ruhr-Universität Bochum

von
Dipl.-Ing. Hans Albers
Marne

Bochum 1976

Ein Beitrag zur digitalen Simulation kontinuierlicher Systeme auf einem Kleinrechner im Hinblick auf die Anwendung für Stromrichterschaltungen

Dissertation
zur Erlangung des Grades eines
Doktor-Ingenieurs
der Abteilung für Elektrotechnik
an der Ruhr-Universität Bochum

von
Dipl.-Ing. Hans Albers
Marne

Bochum 1976

UNIV.
LEIDEN
BIBL.

Dissertation eingereicht am: 28. Mai 1976
Tag der mündlichen Prüfung: 22. Juli 1976
Referent: Prof. Dr.-Ing. M. Depenbrock
Korreferent: Prof. Dr.-Ing. W. Weber

Diese Arbeit entstand während meiner Tätigkeit als Wissenschaftlicher Assistent am Lehrstuhl für Erzeugung und Anwendung elektrischer Energie der Ruhr-Universität Bochum.

Herrn Prof. Dr.-Ing. M. Depenbrock danke ich sehr für die Anregung zu dieser Arbeit und für die freundliche Unterstützung.

Herrn Prof. Dr.-Ing. W. Weber danke ich für das der Arbeit entgegengebrachte Interesse und für die Übernahme des Korreferates.

Inhaltsverzeichnis

		Seite
0.	Formelzeichen und Indizes	7
1.	Einleitung	8
1.1	Aufgabenstellung	9
2.	Beschreibung der Simulationssprache DIGSI	11
2.1	Aufbau und Funktion des Simulationsprogramms	12
2.2	Das Integrationsverfahren	14
2.3	Syntax und Ablochkonventionen	17
2.3.1	Strukturbeschreibung	18
2.3.2	Parameterbeschreibung	19
2.3.3	Anfangswertbeschreibung	20
2.3.4	Bearbeitungsbeschreibung	21
2.4	Fehlerbehandlung	22
2.5	Ergebnisausgabe	23
2.6	Änderungen des Rechenlaufes	23
3.	Die Implementierung des Simulationsprogramms auf dem Prozeßrechner Siemens 305	31
3.1	Gründe für die Implementierung eines Simulationsprogramms auf einem Kleinrechner	31
3.2	Prinzipielle Lösung für die Implementierung des Simulationsprogramms auf dem Prozeßrechner S305	33
3.2.1	Lösung durch Programmkettung	34
3.2.2	Lösung durch Programmüberlagerung	35
3.3	Aufteilung des Gesamtprogramms	37
4.	Ausbau des Simulationsprogramms	39
4.1	Optimierung des Programms	39
4.1.1	Optimierung auf maximalen Problemumfang	40
4.1.2	Optimierung auf minimale Laufzeit	41

		Seite
4.2	Erweiterungen des Simulationsprogramms	42
4.2.1	Vollständigkeit der Grundfunktionstypen	43
4.2.2	Sortierung der Rechenreihenfolge	44
4.2.3	Plotausgabe und Fourieranalyse	46
4.2.4	Dialogbearbeitung und Statiktest	48
4.2.5	Automatische Simulation mit Parametervariation	49
5.	Die digitale Simulation von Netzwerken mit Schalterelementen	51
5.1	Die Simulation von Stromrichterschaltungen durch Netzwerkprogramme	51
5.2	Die Simulation von Stromrichterschaltungen mit allgemeinen Simulationsprogrammen	52
5.2.1	Das Modell einer n-pulsigen Mittelpunktschaltung	54
5.2.2	Stromrichterschaltungen mit parallelen Ventilzweigen innerhalb einer Kommutierungsgruppe	58
5.2.3	Stromrichterschaltungen mit widerstandslosen Maschen	58
5.3	Simulationsbeispiele von Mittelpunktschaltungen	60
5.3.1	Einpulsstromrichter mit Nullanode	60
5.3.2	Der Gleichstromsteller	68
5.4	Die Simulation von Einphasenbrückenschaltungen	72
5.4.1	Die unsymmetrisch halbgesteuerte Brückenschaltung	72
5.4.2	Die löschbare, unsymmetrische Brückenschaltung	79
6.	Zusammenfassung	87
7.	Schrifttum	88
Anhang		
A1	DIGSI-Syntax	90
A2	Die Bestimmung von integralen Meßwerten	91
A3	Daten des Triebzuges ET 420	93
Lebenslauf		95

0 Formelzeichen und Indizes.

Formelzeichen:

a	Fourierkoeffizient
a	Aussteuerung
b	Fourierkoeffizient
C	Kondensator
c	Verstärkungsfaktor
D	Diode, Schaltfunktion der Diode
E	Quellspannung
F	Übertragungsfunktion
f	Funktion
i	Augenblickswert des Stromes
I	Effektivwert des Stromes
L	Induktivität
L	Last
R	Widerstand
S	allgemeine Schaltfunktion
T	Thyristor, Schaltfunktion des Thyristors
T	Zeitkonstante
T1,T2,T3	Rechenschrittweite
t	Zeit
u	Augenblickswert der Spannung
V	Spannung
x	Variable
Z	Impedanz
z	Zündimpulsfunktion

Indizes:

a,1	Netzseite
a	Ausgang
d	Gleichspannungsseite
D	Diode
e	Eingang
i,k,l	ganze Zahl, laufend
N	Nenngröße
q	Grundschwingungsblindanteil
T	Thyristor
w	Wirkgröße

1 Einleitung.

Mit fortschreitender Entwicklung digitaler Rechenanlagen gewinnt die Simulation von Problemen mit Hilfe eines Digitalrechners größere Bedeutung. Zum einen liegt der Grund darin, daß die zu bearbeitenden Probleme umfangreicher und komplexer werden und deshalb mit einem Analogrechner nicht mehr bewältigt werden können. Andererseits ist auch eine Entwicklung allgemeiner und spezieller Simulationssprachen zu sehen, was die Anwendung der digitalen Simulation attraktiver macht.

Für die Simulation von kontinuierlichen, zeitinvarianten Systemen gibt es eine größere Auswahl von Simulationsprogrammen, beispielsweise ASIM, ANAGOL, MIMIC oder CSMP, die in der Regel für große Rechenanlagen konzipiert und teilweise wegen der kompilierenden Arbeitsweise an den Rechnertyp gebunden sind. Die Programme bilden im wesentlichen die Rechenfunktionen des Analogrechners nach und sind dabei ergänzt durch weitere mathematische Funktionen, die sich bei einem Digitalrechner leicht darstellen lassen.

Aufgrund ihres Programmablaufes kann man zwischen den übersetzenden und den interpretierenden Simulationsprogrammen unterscheiden. Bei der ersten Art besteht eine große Freizügigkeit in der Form der Problembeschreibung, die in der Schreibweise der mathematischen Gleichungen geschehen kann, und läßt prinzipiell die Definition zusätzlicher Funktionen zu. Anhand der Problembeschreibung generiert das Simulationsprogramm ein Rechenprogramm, das dann für die eigentliche Simulation abläuft.

Die interpretierenden Simulationsprogramme dagegen erstellen aus der Problembeschreibung, die gleichungs- oder funktionsorientiert sein kann, Verknüpfungs- und Ablauflisten, anhand derer die Simulation durch Berechnung der aufgeführten Funktionen erfolgt. Eine Definition weiterer Funktionen, wie sie durch das Simulationsprogramm vorgegeben sind, ist hierbei meistens nicht zugelassen.

Die entscheidenden Vorteile der digitalen Simulation, die einfache Beschreibung des Problems in der gewohnten, mathe-

matischen Schreibweise und der nahezu unbegrenzte Problemumfang, müssen dabei erkauft werden durch größere Rechenzeiten und längere Auswertung der Ergebnisse, sofern keine zusätzlichen Hilfsmittel der Rechenanlage zur Verfügung stehen.

Die Simulation von Stromrichterschaltungen entspricht wegen der zwei möglichen Ventilzustände einer Darstellung von Schaltungen mit zeitabhängigen Parametern und ist somit von vorneherein mit den vorgenannten Programmen nicht zu bearbeiten. In der Literatur werden dafür spezielle Netzwerkprogramme angegeben /14, 15/, bei denen Ventile als Schaltungselemente zugelassen sind. Als Simulationsergebnis liefern die Programme ausgewählte, zeitliche Verläufe von Strömen und Spannungen aus dem Netzwerk. Daneben lassen sie keine erweiternden Darstellungen, etwa eine digitale Regelung, zu; eine zusätzliche Auswertung der Ergebnisse über eine Programmnahtstelle ist ebenfalls nicht vorgesehen. Diesen Einschränkungen unterliegt ein allgemeines Simulationsprogramm nicht, so daß bei Simulationen, die mehr als nur die reine Darstellung des Netzwerkes enthalten, wieder hierauf zurückgegriffen werden muß.

1.1 Aufgabenstellung.

Zur Darstellung von Stromrichterschaltungen mit allgemeinen Simulationsprogrammen wird für den institutseigenen Prozeßrechner Siemens 305 ein solches Programmsystem implementiert. Als Grundlage dafür dient ein von der Firma BBC speziell entwickeltes Simulationsprogramm DIGSI /4/. Das Programm muß für die Rechenanlage umgearbeitet werden, da hier nur ein Arbeitsspeicher von etwa 11 k Worten zur Verfügung steht. Über eine alphanumerische, graphische Sichtgerätestation soll der Anwender mit dem Programm im Dialog arbeiten können, damit die Vorteile der analogen Simulation auch für das Digitalrechenprogramm Geltung haben:

- enger Kontakt zwischen Benutzer und Rechner,
- Eingriffsmöglichkeiten während des Simulationslaufes,
- unmittelbarer Zugriff und gute Auswertbarkeit der Ergebnisse.

Der verfügbare Arbeitsspeicher soll optimal ausgenutzt werden, so daß ein möglichst großer Problemumfang mit dem Programm rechenbar ist.

Um Stromrichterschaltungen, das heißt Schaltungen mit zeitabhängigen Parametern, mit Hilfe dieses Simulationsprogramms darstellen zu können, sind dafür allgemeingültige Modellgleichungen anzugeben. Am Beispiel der löschbaren, unsymmetrischen Einphasenbrückenschaltung soll die Simulation von Stromrichterschaltungen dargestellt und die Steuerkennlinie bestimmt werden.

2 Beschreibung der Simulationssprache DIGSI.

Die Simulation eines Problems erfordert zunächst dessen mathematische Beschreibung, die die Zusammenhänge zwischen den Zustandsgrößen und einer unabhängigen Variablen, zum Beispiel der Zeit, angibt. Dieses mathematische Modell besteht bei kontinuierlichen Systemen aus einem Satz von Differentialgleichungen. Zur Lösung des Problems mit einem Simulationsprogramm muß das mathematische Modell so abgefaßt werden, daß es von dem Programm interpretierbar ist. Dabei müssen Regeln beachtet werden, die die formale Abfassung der Modellbeschreibung betreffen; sie sind Teil dessen, was im weiteren als Simulationssprache bezeichnet wird. Neben diesen Formalvereinbarungen (Syntax) enthält die Sprache noch die Abgrenzungen zu ihrem mathematischen Inhalt (Semantik).

Bei den übersetzenden Simulationssprachen lassen die Formalregeln eine große Freizügigkeit zu und erlauben eine Problembeschreibung in Form von mathematischen Gleichungen, was eine leichte Handhabung bedeutet. Der Sprachinhalt als Untermenge der Programmiersprache, in der das Programm abgefaßt wurde, ist somit ebenfalls sehr groß und erlaubt beispielsweise die Definition von Funktionen in Makroform.

Interpretierende Simulationsprogramme, zu denen DIGSI gehört, haben dagegen strenge Regeln für die Problembeschreibung und lassen auch keine weitere Definition von zusätzlichen Funktionen zu. Sprache und Programm sind anhand der Grundfunktionen aufgebaut, also blockorientiert. Der Vorteil der blockorientierten Sprachen liegt darin, daß sich Änderungen der Problembeschreibung während des Simulationslaufes oder bei Wiederholungen einfach durchführen lassen und keine vollständig neue Beschreibung erfordern.

Das mathematische Modell des Problems läßt sich in Modellstruktur und Modellparameter einteilen; die Struktur enthält das Gleichungssystem, die Parameter sind Konstante und Startwerte für die Rechnung. Für die digitale Simulation wird noch ein weiterer Datensatz benötigt, der den Ablauf der

Simulation beschreibt, die Bearbeitungsdaten. Sie enthalten die Angaben zu der Rechnung (Verfahren, Schrittweite und Simulationsende) und zu der Ergebnisausgabe. Mit diesen drei Datensätzen wird die Simulationsaufgabe beschrieben.

2.1 Aufbau und Funktion des Simulationsprogramms.

Das Ziel der digitalen Simulation besteht darin, ein gegebenes Gleichungssystem schrittweise zu berechnen und den schrittweisen Werteverlauf der gewünschten Zustandsgrößen zu verfolgen. Bei der vorliegenden Simulationssprache muß das Gleichungssystem dazu bis zu den implementierten Grundfunktionen hin aufgeschlüsselt werden. Die gewünschten Zustandsgrößen stehen dann als Ergebniswerte bestimmter Funktionen zur Verfügung.

Nach dem funktionellen Ablauf läßt sich das Programm in die folgenden vier Abschnitte einteilen:

A Einlesen und Decodierung der Problembeschreibung,
B Berechnung von Parametern,
C Berechnung der Startwerte und
D schrittweise Berechnung der Funktionen mit Ergebnisausgabe.

Diese vier Teile werden von einem Rahmenprogramm gesteuert, das die entsprechenden Entscheidungen dafür trifft, welche Programmteile zu durchlaufen sind, wann es sich um den erstmaligen Start, um eine Korrektur oder um eine Laufmodifikation handelt. Der Datenfluß der einzelnen Programmteile wird über einen gemeinsamen Speicherbereich abgewickelt, in dem die notwendigen Listen dafür aufgebaut sind. Da das Programm in FORTRAN abgefaßt ist, liegen Listengrößen und Speichergrenze fest. Die Zuordnung der Programmteile zu den Listen und der Listen untereinander werden durch einen blockorientierten Aufbau und durch entsprechende Koordinierungslisten erreicht, die die Daten der implementierten Grundfunktionen enthalten. Die Bilder 2.1/1 und 2.1/2 auf Seite 25 und 26 verdeutlichen die Zusammenarbeit zwischen Programmteilen und Datenbereichen; man erkennt, daß aus der Problembeschreibung im Teil A die Datenlisten aufgebaut werden, die dann schließlich im Programm-

teil D zu den Ergebnissen verarbeitet werden. Im folgenden wird der Ablauf einer Simulation, der als Flußdiagramm in Bild 2.1/2 Seite 26 dargestellt ist, anhand der Programmfunktionen erläutert.

Vom Rahmenprogramm wird zuerst der Teil A angesteuert, der die Daten der Modellstruktur, der Parameter, der Anfangswerte und die Bearbeitungsdaten in die zugehörigen Speicherlisten ablegt. Dabei werden die Daten auf ihre formale Richtigkeit geprüft. Die Zwischenspeicherung der Daten ist erforderlich, da über die Reihenfolge der vier Datensätze beliebig verfügt werden kann, und eine Korrektur fehlerhafter Daten ermöglicht werden muß. Außerdem läßt sich dadurch eine Änderung während der Simulationsrechnung oder bei einer Wiederholung leichter realisieren.

Anschließend wird die Liste der Strukturbeschreibung einer Umcodierung unterworfen, wobei die Schrittweite der Simulation eingearbeitet wird. Die daraus entwickelte Reihenfolgeliste enthält Informationen über Funktionstyp, Funktionsnummer, Verweise auf Ausgang und Eingänge und über die Reihenfolge, nach der die Funktionen berechnet werden. Die Verarbeitung der Koordinierungslisten in diesem Programmstadium bewirkt, daß bei der schrittweisen Rechnung ein schnellerer Zugriff auf die variablen Größen erfolgen kann. Aus der Strukturliste hat sich damit die Reihenfolgeliste ergeben.

Im folgenden Abschnitt B des Programms wird anhand der Reihenfolgeliste untersucht, für welche Funktionen Parameter errechnet werden müssen. Sie werden aus den eingegebenen Problemparametern entnommen und mit den Bearbeitungsdaten zu einer neuen Parameterliste errechnet. Fehlende Eingaben werden zu Null angenommen und führen unter Umständen zu Fehlermeldungen. Die Parameter werden so aufbereitet, daß der spätere Rechenlauf zeitoptimal abläuft.

Im Teil C errechnet das Programm die Startwerte aller aufgeführten Funktionen. Da die Anfangswerte der speichernden Funktionen bereits im Teil A des Programmablaufs in den Er-

gebnislisten abgelegt sind, werden hier nur die Startwerte aller übrigen Funktionen nach der Reihenfolgeliste bestimmt. Die Simulation befindet sich danach vergleichend mit dem Analogrechner im Zustand 'Statiktest'. Für die Übertragungsfunktionen werden noch die erforderlichen Zwischenwerte in den Speicherlisten eingetragen.

Die schrittweise Durchrechnung des Problems geschieht durch den Teil D des Programms. Anhand der Reihenfolgeliste werden die benötigten Eingangswerte aus den Ergebnislisten ermittelt und über die Rechenalgorithmen die Ausgangsgröße der Funktion bestimmt und dort abgelegt. Die Ergebnisliste enthält somit nur für jeden Rechenschritt den aktuellen Stand der Simulation. Gewünschte Ergebnisse werden nach Maßgabe der Schrittfolge zusammengestellt und auf dem gewählten Medium ausgegeben.

Im Rahmenprogramm wird nach jedem Rechenschritt geprüft, ob eine der Abbruchbedingungen (Simulationsende, Interrupt oder Konsolschalter) erfüllt ist; der Simulationslauf wird dann unterbrochen.

2.2 Das Integrationsverfahren.

Die digitale Simulation stellt die numerische Lösung des mathematischen Modells dar. Enthält das Modell im allgemeinen Fall integrierende Glieder, so ist die numerische Integration eines Systems von Differentialgleichungen erforderlich. Die Integration der Differentialgleichungen wird bei den bekannten Simulationsprogrammen durch Integration von Differentialgleichungen 1. Ordnung mittels ein- oder mehrschrittiger Verfahren gelöst. Die Lösung der Gleichungen ist dabei je nach Integrationsmethode verschieden ungenau, da sie nur zu Punkten mit endlich großem Abstand (Schrittweite) bestimmt wird.

Im vorliegenden Simulationsprogramm wird ein Integrationsverfahren angewendet, das von der Übertragungsfunktion ausgeht und daraus eine Näherungslösung der Zeitfunktion ermittelt /5/.

Eine lineare Differentialgleichung der Zeit m-ter Ordnung hat eine Laplace-Transformierte

$$F_1(p) = \frac{\sum_{i}^{n} a_i \cdot p^i}{\sum_{j}^{m} b_j \cdot p^j} \quad ; \quad m \geq n \qquad (2.2/1)$$

Diese stellt gleichzeitig die Übertragungsfunktion eines Systems mit m unabhängigen Energiespeichern dar ($m \geq n$). Da die Lösungsfunktion bei der Simulation nur zu bestimmten Zeitpunkten benötigt wird, genügt es, die Differentialgleichung für die Abtastpunkte zu lösen. Die m Energiespeicher lassen sich dann durch m Totzeitglieder mit der Abtastzeit als Totzeit darstellen.

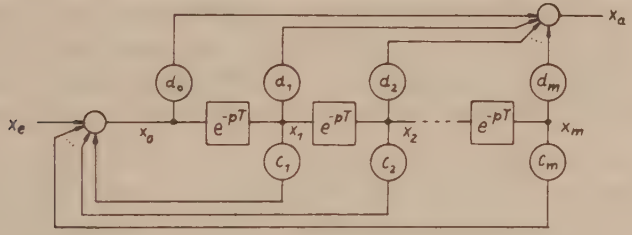


Bild 2.2/1 Blockschaltbild mit m Energiespeicher

Die Übertragungsfunktion zu diesem Blockschaltbild lautet:

$$F_2(p) = \frac{x_a(p)}{x_e(p)} = k\frac{\sum_{k=0}^{m} d_k \cdot e^{-kpT}}{1 - \sum_{l=1}^{m} c_l \cdot e^{-lpT}} \qquad (2.2/2)$$

Mit Hilfe der Transformation $e^{-pT} = (1 - w \cdot \frac{T}{2}) / (1 + w \cdot \frac{T}{2})$ läßt sich $F_2(p)$ so umformen und normieren, daß eine eindeutige Bestimmung der Koeffizienten von $F_2(p)$, d_k und c_l, aus denen von $F_1(p)$, a_i und b_j, möglich ist.

Die Ausgangsgröße x_a ergibt sich somit für jeden Abtastschritt (= Rechenschritt) aus der Eingangsgröße x_e und den gespeicherten Werten x_i mittels der Koeffizienten d_i und c_i.

$$x_0(n) = x_e(n) + \sum_{i=1}^{m} c_i \cdot x_i(n)$$

$$x_a(n) = \sum_{i=0}^{m} d_i \cdot x_i(n) \qquad (2.2/3)$$

Danach werden die Speicher entsprechend der Totzeitvorschrift aktualisiert.

$$x_i(n+1) = x_{i-1}(n) \qquad (2.2/4)$$

Das Integrationsverfahren benötigt zur Lösung der Differentialgleichung für jede Funktion 2m+1 Speicherplätze, und zwar für die Koeffizienten und die Zwischenspeicher. Daher sind für das Simulationsprogramm nur Übertragungsfunktionen 1. und 2. Ordnung vorgesehen; die Funktionen höherer Ordnung können durch Reihenschaltung dieser beiden ersetzt werden.

Das Integrationsverfahren liefert nicht den Momentanwert der Lösungsfunktion, sondern den Integralwert bezogen auf das Rechenintervall. Daraus resultiert, daß das Verfahren auch bei großer Schrittweite stabile Lösungen aufweist, sofern die Übertragungsfunktion Tiefpaßverhalten hat.

Der Vergleich mit Integrationsverfahren anderer Simulationsprogramme stellt sehr günstige Ergebnisse heraus /6/. Der Integrationsfehler liegt dabei in derselben Größenordnung; die erforderliche Rechenzeit liegt zum Teil sogar niedriger.

Wie bei mehrschrittigen Integrationsverfahren ist auch dieses nicht selbststartend. Die Zustandsgrößen x_i müssen dafür in der Anfangswertrechnung aus der vorliegenden Eingangsgröße und dem Startwert der Funktion bestimmt werden.

Hinsichtlich des Programmaufbaus bietet das Verfahren den Vorteil, daß mit einer konstanten Schrittweite gerechnet wird. Dadurch ergeben sich äquidistante Lösungspunkte, und es entfällt die Bestimmung nicht berechneter Zwischenwerte.

2.3 Syntax und Ablochkonventionen.

Da hier eine blockorientierte Sprache vorliegt, gelten nur wenige syntaktische Regeln, die im Anhang in der Backus- Notation zusammengefaßt sind. Da die Problembeschreibung über FORTRAN-Leseanweisungen formatgebunden übermittelt wird, unterliegt die Form der Abfassung strengen Vereinbarungen. Diese Regeln sind bei der Problembeschreibung zu beachten und werden in diesem Abschnitt näher erläutert.

Das Einlesen der Problembeschreibung erfolgt in vier Gruppen, die jeweils die Daten der Struktur, der Parameter und der Anfangswerte sowie die Bearbeitungsdaten beinhalten. Es wird zeilenweise gelesen, wobei nur die ersten 69 Zeichen signifikant sind. Die Eingabegruppen sind in ihrer Reihenfolge beliebig und können wiederholt und auch ergänzt werden, solange das Endekennzeichen der Datenübermittlung noch nicht gegeben wurde. Zur Kennzeichnung der Gruppen dienen die Anfangsbuchstaben ihrer alphanumerischen Zeichenfolge, vorangestellt durch zwei Zeichen '+'[1].

++STRUKTUR	Gruppe der Strukturdaten
++PARAMETER	Gruppe der Parameterdaten
++ANFANGSWERTE	Gruppe der Anfangswerte
++BEARBEITUNG	Gruppe der Bearbeitungsdaten
EN	Gruppenendekennzeichen
FINIS	Ende der Datenübermittlung

Die Unterstreichung gibt den Teil der Zeichenfolge an, der für die Identifizierung relevant ist.

Das Ende einer Datengruppe wird durch die Zeichen 'EN' angegeben, das Ende der gesamten Daten durch 'FI'.

Zur Kennzeichnung der zu übermittelnden Daten werden alphanumerische Kurzzeichen benutzt, die in bestimmter Reihenfolge abgelocht sein müssen. Zu deren Beschreibung wird definiert:

---

1) Das Zeichen '+' steht hier für das Zeichen 'Stern'.

tt	Grundfunktion
nn	Nummer einer Grundfunktion
sw	Definition der Schrittweite
z	reelle Zahl
k	Ziffer
x	alphanumerisches Zeichen
b	beliebiges Trennungszeichen
_	Leerzeichen

Die in dem Simulationsprogramm implementierten Grundfunktionen sind die Rechenoperationen, die von dem Programm für die Problemrechnung ausgeführt werden können. Sie werden zur Definition mnemotechnisch abgekürzt durch zwei alphanumerische Zeichen (zum Beispiel: AD = Addierer) und nur intern nach arithmetischen und logischen Funktionen aufgeschlüsselt geführt. Da für jede Grundfunktion Listenbereiche vereinbart sind, wird zur eindeutigen Identifizierung eine zweistellige Zahl mitgeführt, so daß eine Funktionsbezeichnung durch diese beiden Merkmale spezifiziert ist.

Funktion ::= ttnn

Die Anzahl der implementierten Grundfunktionen, der Funktionstyp und ihre Maximalzahl sind aus der Tabelle 2.3/1 Seite 27 zu entnehmen.

Konstanten und Parametern werden die Zahlenwerte ausschließlich als reelle Zahlen zugewiesen.

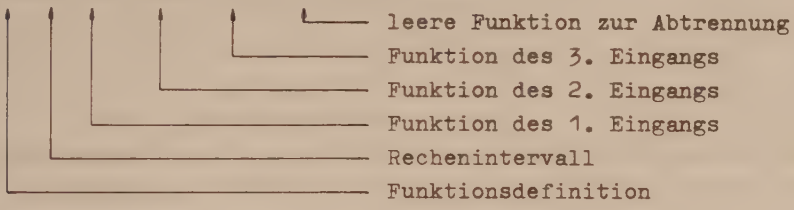
2.3.1 Strukturbeschreibung.

In dieser Gruppe der Problembeschreibung werden für jede Funktion die Verknüpfungen der Eingänge mit Ausgängen anderer Funktionen und ihr Rechenintervall aufgeführt. Um den Zusammenhang der Funktionen untereinander leichter zu erkennen, entwickelt man dazu günstigerweise aus dem mathematischen Modell ein Blockschaltbild mit den Wirkungslinien, wie es vom Analogrechner her geläufig ist. Gleichzeitig erhält man Aufschluß darüber, bis zu welchen Grundfunktionen das Modell aufgeschlüsselt werden muß, welche Funktionen verwendet werden

und welche Verbindungen notwendig und möglich sind.

Da alle Grundfunktionen nur einen Ausgang haben, brauchen sie nicht gesondert definiert zu werden; die Funktionsdefinition erfolgt zusammen mit den anderen Angaben. Eine Strukturanweisung enthält in einer Zeile die Funktionsdefinition, das Rechenintervall und die Eingangsverbindungen zu den anderen Funktionen. Die Zeile hat damit folgende Zeichenfolge:

ttnnswttnnbttnnbttnnb----bbb....



Beispiel:
AD01+ SU13,EK10,U123,

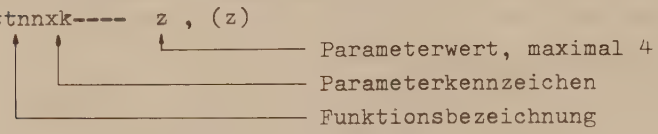
Da aufgrund des Integrationsverfahrens mit konstanter Schrittweite gerechnet wird, enthält die Intervallangabe 'sw' eine symbolische Kennzeichnung für eine der drei möglichen Schrittweiten, in der die Funktion gerechnet werden soll; diese Schrittweite wird unter den Bearbeitungsdaten mit einem Wert spezifiziert.

Die Minimalangaben einer Strukturzeile sind Definition, Intervall und der erste Eingang; folgende Eingänge werden automatisch mit Null belegt. Sollen beginnende Eingänge zu Null gesetzt werden, so wird der Eingang mit einer Konstanten 'EK' verbunden, die den Wert Null hat.

2.3.2 Parameterbeschreibung.

Einige Funktionen erfordern zur Berechnung zusätzlich konstante Größen, die innerhalb eines Simulationslaufes nicht verändert werden. Diese Werte werden als Parameter zu den Funktionen übergeben. Die Anzahl der Parameter einer Funktion hängt von ihrem Typ und von der Problemstellung ab. Die Tabelle 2.3/1 S.28 gibt Aufschluß darüber,welche Grundfunktionen Parameter erfordern, wieviele nötig sind, und welche Bedeutung sie haben.

Die Zuordnung der Werte an die Funktion erfolgt mittels der Funktionsbenennung und in festgelegter Reihenfolge. Bei mehreren Parametern dient ein zusätzliches Kennzeichen zur Einordnung. Die Zahlenwerte werden als reelle Zahlen in einem Feld von zehn Zeichen angegeben. Eine Parameteranweisung hat die Form:



Beispiel:

```
FG02 3    -20.6     5.0
```

Das zusätzliche Kennzeichen hat allein bei den Übertragungsfunktionen und beim Funktionsgenerator Bedeutung, und zwar mit folgenden Definitionen.

```
x::= z            Zählerpolynom
x::= n            Nennerpolynom
k::= 1,2,3,4,5    Zeilennummer
```

Die Übertragungsfunktionen verlangen das Zähler- und Nennerpolynom unmittelbar nacheinander, beim Funktionsgenerator wird mit je zwei Stützpunkten eine aufsteigende Zeilennummer erwartet.

2.3.3 Anfangswertbeschreibung.

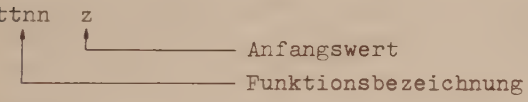
Unter Anfangswerte werden die Daten verstanden, die die Startrechnung der Simulation entscheiden; sie werden deshalb von den eigentlichen Parametern abgetrennt.

Bevor die aufgeführten Funktionen nach Maßgabe der Reihenfolgeliste schrittweise gerechnet werden, ermittelt die Anfangsrechnung die Vorgeschichte der Funktionen. Diese wird allein durch die Konstanten und durch die Anfangswerte der speichernden Funktionen bestimmt, deren Werte in dieser Datengruppe in eine Anfangswertliste eingeschrieben werden.

Mit dem Endekennzeichen der Datengruppe wird die Simulation auf den Anfangszustand gesetzt. Das beinhaltet das Rücksetzen von Dateien, das Löschen der Ergebnislisten und die Übernahme

der Anfangswerte aus ihrer Liste in die Ergebnislisten. Deshalb muß diese Datengruppe auch dann aufgerufen werden, wenn die Simulation keine Anfangswerte erfordert; die Anfangswertliste ist dann Null.

In der Tabelle 2.3/1 sind die Grundfunktionen gekennzeichnet, zu denen Anfangswerte angegeben werden können. Die Zuordnung wird auch hier durch die Funktionsbezeichnung erreicht. Als Zahlenangabe sind nur reelle Zahlen zugelassen. Eine Zeile der Anfangswertbeschreibung lautet somit:



Beispiel:

```
TZ04   -14.5
```

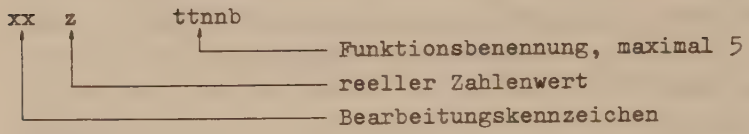
Die Anfangswertliste ist begrenzt auf 70 Werte, da sie nur die aufzunehmen braucht, die ungleich Null sind.

2.3.4 Bearbeitungsbeschreibung.

Neben der eigentlichen Problembeschreibung benötigt das Programm zur Simulation noch die Daten, die den Ablauf steuern. Es sind dies die Angaben zu der Integrationsschrittweite, zur Simulationsdauer und zur Ausgabe der Ergebnisse, die tabellarisch ausgedruckt werden können oder auf einen Massenspeicher abgelegt werden. Weiterhin kann noch eine Unterdrückung der Sortierung von der Reihenfolgeliste veranlaßt werden.

Die Schrittweite, die in der Strukturbeschreibung symbolisch für jede Funktion angegeben wird, erhält durch die Angabe der Rechenintervalle (T1, T2, T3) konkrete Zahlenwerte. Dabei ist die Bedingung $T1 \leq (T2, T3)$ aus rechentechnischen Gründen einzuhalten. Bei der tabellarischen Ausgabe der Ergebnisse können zwei Ausgabeintervalle angeführt werden, die durch einen Grenzwert voneinander getrennt werden. Dadurch lassen sich interessierende, dynamische Vorgänge detailliert ausgeben und stationäre in größeren Abständen. Die möglichen Bearbeitungsanweisungen und deren Bedeutung sind in der Tabelle 2.3.4/1 Seite 29 angeführt.

Die Zeile einer Bearbeitungsanweisung hat die formale Form:



Beispiele:

```
KU   7.0    AD17,FG04
T1   3.5
```

Das Programm prüft dabei nicht, ob alle notwendigen Anweisungen aufgeführt sind; durch entsprechende Voreinstellungen wird ein Fehllauf vermieden.

2.4 Fehlerbehandlung.

Die Problembeschreibung erfolgt zeilenweise, so daß jede Zeile für sich geschlossen interpretierbar ist. Daher prüft das Programm jede Anweisungszeile auf ihre formale Richtigkeit, das heißt, auf ihre Interpretierbarkeit. Sind dagegen Daten fehlerhaft abgelocht, so führt das wegen der formatgebundenen FORTRAN-Leseanweisung zu einem Programmabbruch; eine Überprüfung der Datenkarten wäre dann erforderlich.

Kann das Programm während der Auswertung einer Zeile Zeichen nicht einordnen, so gibt es eine Fehlermeldung ab und unterbricht die weitere Interpretation der Zeile. Zugleich wird die fehlerhafte Zeile protokolliert und an der Stelle, an der dieser Fehler entdeckt wurde, ein Fehlerkurzzeichen darunter gesetzt. Aus der Tabelle 2.4/1 auf Seite 30 läßt sich mit Hilfe der Kurzzeichen auf den Fehlertyp schließen.

Ist die Problembeschreibung abgeschlossen, und stellt das Programm fest, daß mindestens ein Fehler aufgetreten ist, so fordert es mit einer Meldung zur Korrektur auf. Korrigierte Zeilen können eingegeben werden. Das Programm registriert jedoch nicht, ob alle Fehler dabei beseitigt wurden.

2.5 Ergebnisausgabe.

Das Ziel einer Simulation ist der Verlauf gewählter Zustandsgrößen in Abhängigkeit von der unabhängigen Variablen oder die Werte signifikanter Größen. Diese stehen bei der blockorientierten Simulation als Ausgänge entsprechender Funktionen zur Verfügung und können durch Angabe in den Bearbeitungsanweisungen ausgewählt werden. Zusammen mit der unabhängigen Variablen kann man die Werte der Funktionen tabellarisch ausgeben oder auch auf einem Massenspeicher hinterlegen lassen. Die weitere Auswertung der Ergebnistabelle ist dann dem Anwender überlassen; die abgespeicherten Daten stehen anderen Auswerteprogrammen zur Verfügung. Der Datenbereich bleibt solange erhalten, bis er bei einem neuen Rechenlauf überschrieben wird.

2.6 Änderungen des Rechenlaufes.

Wenn bei der schrittweisen Durchrechnung der Problemstruktur die unabhängige Variable den Maximalwert erreicht hat, der bei den Bearbeitungsdaten als Simulationsdauer angegeben ist, so ist damit der normale Simulationslauf beendet. Das Programm kann abgebrochen werden, wobei die gespeicherten Daten zur weiteren Auswertung erhalten bleiben, oder es kann zu einem Wiederholungslauf angestoßen werden.

Der Wiederholungslauf erfordert das Rücksetzen der Simulation in den Anfangszustand, was mit der Eingabe von Anfangswerten geschieht. Es werden dabei nur die Anfangswerte eingegeben, die gegenüber dem alten Zustand geändert werden sollen. Bei einem einfachen Wiederholungslauf entsprechend dem repitierenden Betrieb beim Analogrechner ist die neue Anfangswertliste leer.

Die Konstruktion als blockorientiertes Programm gestattet eine beliebige Änderung der Problembeschreibung innerhalb eines Simulationslaufes. Der Programmlauf kann dazu entweder durch äußeren Eingriff über einen Konsolschalter oder vom Problemlauf über die Interruptfunktion 'IT' angehalten werden. Die Unterbrechung erfolgt nach dem Ende des gerade laufenden

Rechenschrittes. Die Abspaltung der Anfangswerte von den Problemparametern wirkt sich hierbei günstig aus, weil dadurch Parameter abgeändert werden können, ohne gleichzeitig ein Rücksetzen der Simulation in den Anfangszustand zu bewirken.

Mit dem Endekennzeichen der Datenübergabe wird die Fortsetzung des Rechenlaufes angestoßen.

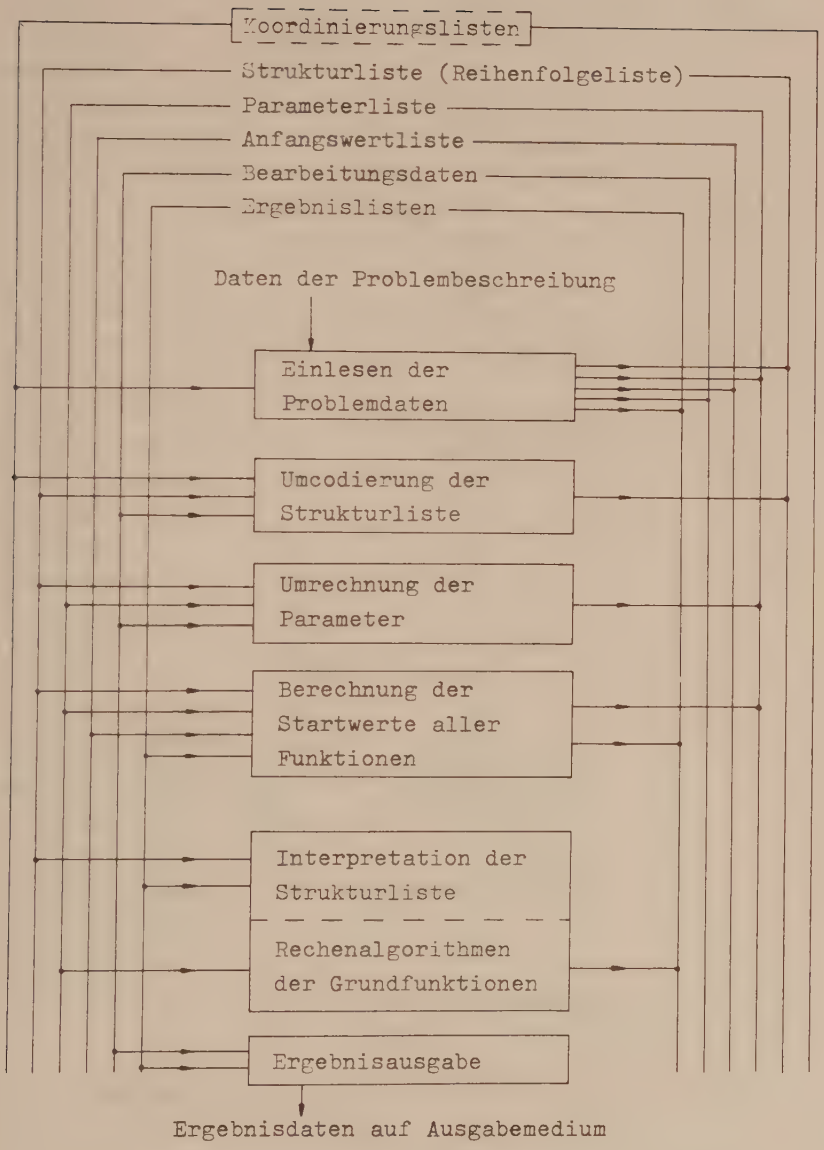


Bild 2.1/1 Programmkopplung über die Listenführung

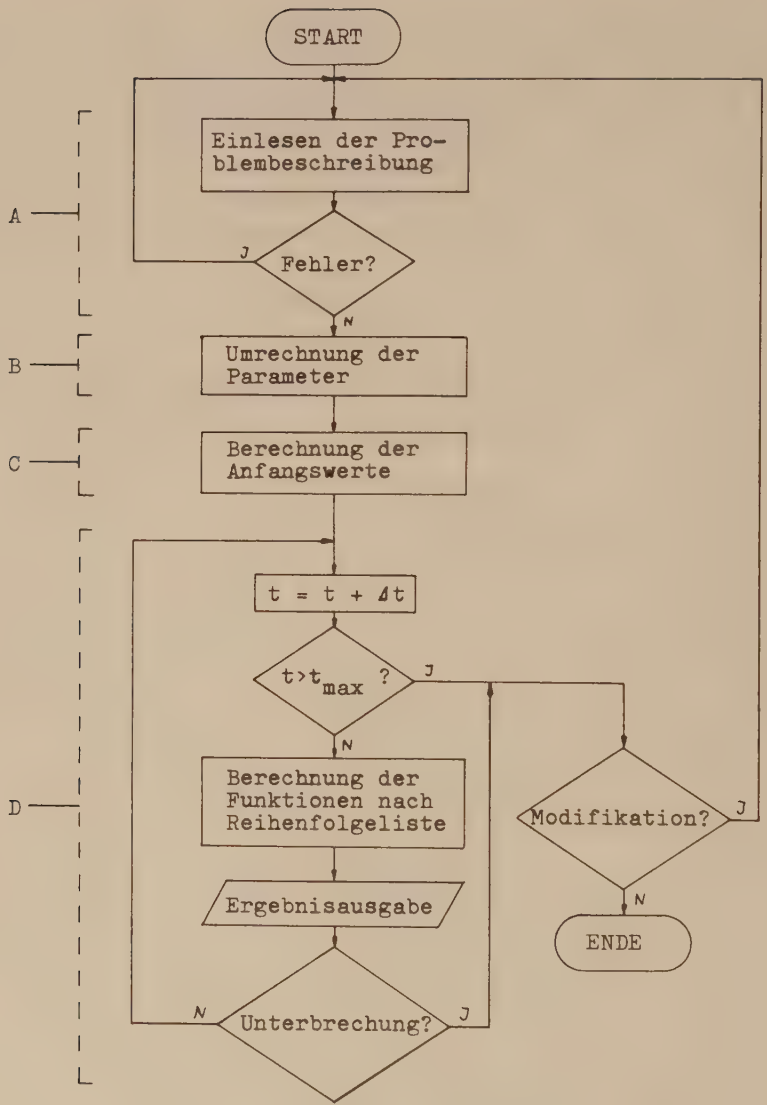


Bild 2.1/2 Programmablauf

Funktion	Benennung	Typ	Anzahl	Eing.-Typ 1. 2. 3.	Startwert
Addierer	AD	a	70	a a a	-
Summierer	SU	a	70	a a -	-
Verstärker	UO	a	70	a - -	-
Funktion 1. Ordg.	U1	a	30	a - -	ja
Funktion 2. Ordg.	U2	a	20	a - -	ja
Totzeit	TZ	a	10	a - -	ja
Konstante	EK	a	40	- - -	ja
Wurzelfunktion	WU	a	10	a - -	-
Modulofunktion	MO	a	10	a a -	-
Sinusfunktion	SI	a	10	a - -	-
Absolutfunktion	AB	a	15	a - -	-
Integerfunktion	IN	a	10	a - -	-
Funktionsgenerator	FG	a	5	a - -	-
Multiplizierer	MU	a	20	a a -	-
Dividierer	DI	a	20	a a -	-
Begrenzer	BG	a	15	a - -	-
D/A-Schalter	SW	a	25	1 a a	-
Track-Store-Fkt.	TS	a	15	1 a -	-
Komparator	KO	1	30	a a -	-
Flip-Flop	FF	1	25	1 1 1	ja
Monoflop	MF	1	5	1 - -	-
logisches UND	UN	1	30	1 1 -	-
logisches ODER	OR	1	30	1 1 -	-
logisches NICHT	NI	1	15	1 - -	-
logisches NAND	NA	1	8	1 1 -	-
logisches NOR	NO	1	7	1 1 -	-
Zeitgeber	TM	a	-	- - -	ja
Taktgeber	TA	1	-	- - -	-
Interrupt	IT	1	2	1 - -	-

a . . . analog
1 . . . logisch

Tabelle 2.3/1 Implementierte Grundfunktionen in DIGSI
(Fortsetzung siehe Seite 28)

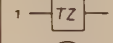
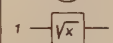
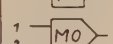
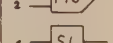
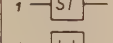
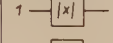
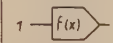
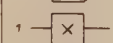
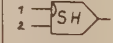
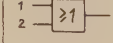
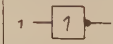
Funktion	Parameter			x Symbol y	Definition
	Zahl	Zahl/Zeile	Reihenfolge		
AD	-	-	-		$y = x_1 + x_2 + x_3$
SU	-	-	-		$y = x_2 - x_1$
UO	1	1	k		$y = k \cdot x_1$
U1	4	2	a,b/c,d		$y(p) = \frac{a + b \cdot p}{c + d \cdot p} \cdot x_1(p)$
U2	6	3	a,b,c/ d,e,f		$y(p) = \frac{a + b \cdot p + c \cdot p^2}{d + e \cdot p + f \cdot p^2} \cdot x_1(p)$
TZ	1	1	T_1		$y(p) = exp(-p \cdot T_1) \cdot x_1(p)$
EK	-	-	-		$y = K$
WU	-	-	-		$y = x_1 \cdot sign(x_1)$
MO	-	-	-		$y = mod(x_1, x_2)$
SI	-	-	-		$y = sin(x_1)$
AB	-	-	-		$y = \lvert x_1 \rvert$
IN	-	-	-		$y = int(x_1)$
FG	20	4	$x_i, y_i, x_{i+1}, \cdots$		$y = f(x_1)$
MU	-	-	-		$y = x_1 \cdot x_2$
DI	-	-	-		$y = x_1 / x_2$
BG	2	2	x_o, x_u		$y = \left\{ \begin{matrix} x_o \\ x_1 \\ x_u \end{matrix} \right. \wedge \left. \begin{matrix} x_1 \geq x_o \\ x_1 \leq x_u \end{matrix} \right\}$
SW	-	-	-		$y = \left\{ \begin{matrix} x_2 \\ x_3 \end{matrix} \right. \wedge \left. \begin{matrix} x_1 = 0 \\ x_1 = L \end{matrix} \right\}$
TS	-	-	-		$y = \left\{ \begin{matrix} x_2 \\ y(t-T_1) \end{matrix} \right. \wedge \left. \begin{matrix} x_1 = L \\ x_1 = 0 \end{matrix} \right\}$
KO	-	-	-		$y = \left\{ \begin{matrix} 0 \\ L \end{matrix} \right. \wedge \left. \begin{matrix} x_1 < x_2 \\ x_1 \geq x_2 \end{matrix} \right\}$
FF	-	-	-		$y = \left\{ \begin{matrix} L \\ 0 \end{matrix} \right. \wedge \left. \begin{matrix} x_1 = L \vee y(t-T_1) = L \wedge x_2 = 0 \\ x_1 = 0 \wedge [x_2 = L \vee y(t-T_1) = 0] \end{matrix} \right\}$ x_3: Takt
MF	1	1	$k = T/T_1$		$y = \left\{ \begin{matrix} L \\ 0 \end{matrix} \right. \begin{matrix} t_1 \leq t < t_1 + T \\ sonst \end{matrix} \left. \right\}$ $t_1 = t$; $x_1 = 0 \swarrow L$
UN	-	-	-		$y = x_1 \wedge x_2$
OR	-	-	-		$y = x_1 \vee x_2$
NI	-	-	-		$y = \overline{x_1}$
NA	-	-	-		$y = \overline{x_1 \wedge x_2}$
NO	-	-	-		$y = \overline{x_1 \vee x_2}$
TM	-	-	-		$y = t$
TA	-	-	-		$y = \left\{ \begin{matrix} L \\ 0 \end{matrix} \right. \wedge t = \left\{ \begin{matrix} 2n \cdot T_1 \\ (2n-1) \cdot T_1 \end{matrix} \right.$
IT	-	-	-		Programmabbruch $x_1 = L$

Tabelle 2.3/1 Implementierte Grundfunktionen in DIGSI (Fortsetzung)

Bezeichnung	Kenn-zeichen	Zahlen-angabe	Vorein-stellung	Bemerkung
Schrittweite	T1	$> 10^{-4}$	1.0	-- } Kennung
Schrittweite	T2	$\geq$ T1	1.0	+- } in der
Schrittweite	T3	$\geq$ T1	1.0	++ } Struktur
Simulationsdauer	SD	$> 10^{-4}$	1.0	
Tabellenausgabe	AU		keine	$\leq$ 5 Funktionen
unteres Intervall	PU	k·T1	1.0	für tab. Ausgabe
oberes Intervall	PO	k·T1	1.0	für tab. Ausgabe
Intervallgrenze	PM	k·T1	1.0	für tab. Ausgabe
Speicherung der Ergebnisse	KU	k·T1	keine	$\leq$ 5 Funktionen
keine Sortierung	NS	0		nicht sortieren
		$\neq$0	$\neq$0	sortieren

Tabelle 2.3.4/1 Bearbeitungsanweisungen

Kennung	Bedeutung
SK	falsche Steuerkarte, keine ++Karte
GA	falscher Gruppenanfang
ND	nicht implementierte Funktion
UE	Überlauf bei der Funktionsnummer
KA	falsche oder keine Anfangsbedingung
KP	falscher oder kein Parameter
PA	formal falsche Parameterzuweisung
AZ	zu viele Funktionen für die Ausgabe
SB	falsche Strukturbeschreibung
BZ	zu viele Strukturbeschreibungen
EI	zu viele oder keine Eingangsverknüpfungen
GZ	Rechenintervall nicht feststellbar
VA	Schrittweite zu klein für U1- oder U2-Funktion
ZG	Stützpunkte beim Funktionsgenerator falsch
DI	Quotient beim Dividierer $>10^{20}$
SB	Funktion nicht rechenbar

Tabelle 2.4/1 Fehlerkennzeichen

3. Die Implementierung des Simulationsprogramms auf dem Prozeßrechner Siemens 305.

Digitale Simulationsprogramme sind in der Regel für große Rechenanlagen konzipiert, weil dort ein großer Arbeitsspeicher verfügbar ist, und kurze Rechenzeiten erwartet werden können. Da man keine Rücksicht auf die Programmlänge nehmen muß, handelt es sich dabei in der Regel um übersetzende Simulationsprogramme, weil diese eine sehr bequeme Benutzung durch den Anwender erlauben.

Bei kleineren Rechenanlagen muß in Hinblick auf die nutzbare Arbeitsspeichergröße auf Programme mit interpretierendem Charakter ausgewichen werden, weil man diese straffer formulieren kann. Dabei muß von daher schon mit längeren Programmlaufzeiten gerechnet werden.

Das vorgestellte, ursprüngliche Programm DIGSI benötigt in seiner einfachsten Form bereits einen Arbeitsspeicher von ca. 20 k Worten; über die benötigte Programmlaufzeit läßt sich keine Vorrausangabe machen, da sie vom Rechnertyp und, wie noch erläutert wird, vom Übersetzertyp abhängt.

Im weiteren werden Methoden angegeben, nach denen große Programme für den verfügbaren Arbeitsspeicher passend umorganisiert werden können. Zuvor werden einige Gesichtspunkte aufgeführt, die eine Implementierung eines derartigen Simulationsprogramms auf einer relativ kleinen Rechenanlage rechtfertigen.

3.1 Gründe für die Implementierung eines Simulationsprogramms auf einem Kleinrechner.

Mit zunehmendem Benutzungsgrad, durch Aufteilung der Rechenaufträge in Kurz- und Langläufe oder durch Einordnung in eine Vorranghierarchie wird die Nutzung eines Simulationsprogramms auf einer Großrechenanlage in den Hintergrund gedrängt, und es wird deshalb für den Anwender wegen der geringeren Verfügbarkeit weniger attraktiv. Das allein wäre schon Grund genug, Simulationsprogramme für kleinere Rechner zu konzipieren, bei denen in der Regel für jeden einzelnen Auftrag mehr Rechenzeit zur Verfügung steht.

Wegen des großen Speicherbedarfs durch das Programm kann wie beim Analogrechner zur gleichen Zeit nur ein Benutzer damit arbeiten; dennoch bleiben die Vorteile der digitalen Simulation erhalten, die mit den folgenden Stichworten umrissen seien:

- großer rechenbarer Problemumfang,
- Vielzahl von Funktionen,
- leichte Erlernbarkeit, keine Normierung,
- nahezu unbeschränkter Wertebereich,
- erhöhte Genauigkeit.

Beim Übergang auf kleinere Rechenanlagen müssen jedoch einige Einschränkungen in Kauf genommen werden, die den rechenbaren Problemumfang, die Anzahl der Funktionstypen und die Rechengenauigkeit betreffen. Das folgt einerseits aus der geringeren Arbeitsspeichergröße und speziell bei der Genauigkeit aus der kürzeren Wortlänge der Rechenanlage. Dafür ergeben sich einige Vorteile, die durch den open-shop-Betrieb bedingt sind. Prioritätsprobleme und Warteschlangen bei den Externelementen (Drucker, Plotter) verlängern hier nicht den Simulationslauf. Weiterhin läßt sich ein direktes Dialogsystem mit dem Programm aufbauen, das die Abwicklung einer Aufgabe erleichtert und einen unmittelbaren Zugriff zu den Ergebnissen gestattet.

Wenn die Erhöhung des Rechenzeitbedarfs nicht berücksichtigt wird, so hängt die Entscheidung für die Implementierung auf einem kleineren Rechner davon ab, in welchem Maße Abstriche am rechenbaren Problemumfang und an den Funktionstypen gemacht werden müssen.

Mit dem ursprünglichen Simulationsprogramm in der Minimalform lassen sich 150 Funktionen rechnen. Dagegen konnte die endgültige Fassung für den Prozeßrechner Siemens 305 auf einen größeren Problemumfang bis zu 250 Funktionen erweitert, und dabei der Vorrat an Funktionstypen ergänzt werden /16/.

3.2 Prinzipielle Lösung für die Implementierung des Simulationsprogramms auf dem Prozeßrechner S 305.

Das ursprüngliche Simulationsprogramm benötigt in der Minimalfassung einen Arbeitsspeicher von ca. 20 k Worten. Bei dem Prozeßrechner steht jedoch nur ein Bereich von 11 k Worten zur Verfügung. Das Hauptproblem besteht somit darin, das Programm so umzugestalten, daß es innerhalb des verfügbaren Speicherplatzes ablaufen kann. Als weiteres Problem tritt hinzu, daß Einbußen am rechenbaren Problemumfang und Verlängerungen der Problemrechenzeiten vermieden werden sollen.

Die Lösung besteht darin, das Programm in genügend kleine Teile aufzuspalten, die dann durch ein geeignetes Betriebssystem oder durch ein spezielles Steuerprogramm in den Arbeitsspeicher geladen werden und dort ablaufen können. Alle Teile werden dabei über einen gemeinsamen Datenbereich koordiniert. Die Zahl der Programmteile hängt dann ab von der Länge des Gesamtprogramms, vom gemeinsamen Datenbereich, von dem verfügbaren Arbeitsspeicher und von der sinnvollen Aufteilung des gesamten Programms.

Die Programmteile werden von einem Massenspeicher zugeladen, der einen schnellen Zugriff erlaubt, wie zum Beispiel magnetische Externspeicher; Papierdatenträger scheiden wegen der niedrigen Lesegeschwindigkeit aus. Wenn die Ablaufreihenfolge der Programmteile nicht sequentiell ist, so muß ein Massenspeicher mit wahlfreiem Zugriff gefordert werden, denn andere Typen (Bandgeräte) müssen wegen der erhöhten Suchlaufzeiten ausgeschlossen werden.

An die Rechenanlage sind daher folgende Forderungen zu stellen: das Betriebssystem muß das Zuladen der Programmteile erlauben, und es muß ein Speicher der Qualität Platten- oder Trommelspeicher vorhanden sein.

Die so skizzierte Ablaufsteuerung bereitet jedoch Schwierigkeiten bei solchen Programmen, die in einer höheren Programmiersprache abgefaßt sind, wie beispielsweise FORTRAN. Die Übersetzung eines selbständigen FORTRAN-Programms bringt zunächst ein abgemagertes Maschinenprogramm hervor, bei dem nur die einfachsten Anweisungen aus dem Quellprogramm direkt in

die Maschinensprache übersetzt sind, während alle anderen durch Unterprogrammaufrufe ersetzt werden. Diese Unterprogramme fügt der Kompiler in einem anschließenden Bindelauf aus der Kompilerbibliothek hinzu.

Sollen die einzelnen Teilprogramme des Simulationsprogramms als selbständige Programmeinheiten abgefaßt werden, so müssen zu der übersetzten Programmlänge noch die erforderlichen Bibliotheksanteile hinzugezählt werden. Da für die Teilprogramme nur ein bestimmter Platz im Arbeitsspeicher zur Verfügung steht, kann man die Programmeinheiten nur schwer anpassen, denn einer Änderung der FORTRAN-Quelle folgt meistens eine andere Zusammenstellung der Bibliotheksteile.

Daher wird eine Lösung angestrebt, bei der alle benötigten Bibliotheksprogramme gesammelt und zu einem gesonderten Programmteil zusammengefaßt sind, auf den dann alle Teilprogramme zugreifen können. Diese bestehen nur noch aus der übersetzten FORTRAN-Quelle, wodurch eine Anpassung an den verfügbaren Platz leichter ist.

Für die weiteren Überlegungen wird vorrausgesetzt, daß das Quellprogramm in FORTRAN abgefaßt ist, und daß der Ablauf der Teilprogramme nicht sequentiell erfolgt, das heißt, erst im Programmlauf selbst wird entschieden, welche Teilprogramme geladen werden müssen. Es ergeben sich von daher zwei mögliche Lösungen.

3.2.1 Lösung durch Programmkettung.

Das Gesamtprogramm wird aufgeteilt in ein Hauptprogramm und eine Reihe von unabhängigen Unterprogrammen (Segmente). Das Hauptprogramm enthält die formalen Aufrufe der Segmente, den erforderlichen, koordinierenden Rechnungsteil und die Deklaration des gemeinsamen Datenbereiches, über den die Datensteuerung mit den Segmenten abgewickelt wird. Die erforderliche Kompilerbibliothek ist ebenfalls dem Hauptprogramm zugeordnet.

Alle Segmente belegen denselben Bereich im Arbeitsspeicher, wobei nur dasjenige Segment geladen ist, das gerade vom Hauptprogramm aufgerufen wurde. Für das Zuladen der Segmente sorgt das Betriebssystem oder ein entsprechendes Steuerprogramm. Die zugehörige Speicherplatzbelegung zeigt im Prinzip das Bild 3.2.1/1.

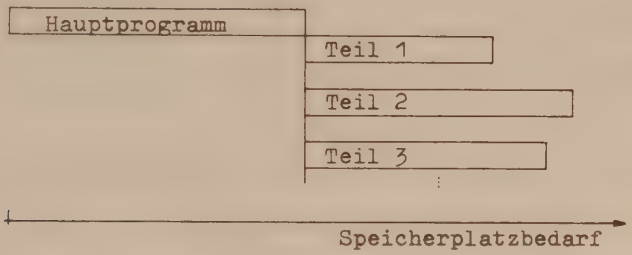


Bild 3.2.1/1 Programmkettung

Soll in der Speicherplatzverteilung das Programm wenig Raum einnehmen, so muß das Gesamtprogramm in viele Teilprogramme zerlegt werden. Das wiederum erfordert ein umfangreiches Hauptprogramm mit einem großen Organisationsteil, da ein sequentieller Ablauf nicht vorrausgesetzt werden kann.

3.2.2 Lösung durch Programmüberlagerung.

Diese Programmiertechnik, auch mit Overlay bezeichnet, gestattet eine Einteilung der Programmsegmente hinsichtlich ihrer Aufrufkompetenz; das Gesamtprogramm ist dabei wie im Abschnitt 3.2.1 in Segmente zerlegt. Den Segmenten können nun Rangordnungen zugeteilt werden, wodurch sich eine baumartige Struktur im Programmaufbau ergibt. In Bild 3.2.2/1 ist dieser prinzipielle Aufbau dargestellt.

Neben der Rangordnung, dargestellt durch unterschiedliche Schraffur, ist dazu die Platzverteilung angegeben, die die einzelnen Programmteile im Arbeitsspeicher einnehmen würden. Das Hauptprogramm nimmt direkt nur Zugriff zu Teilprogrammen der Rangordnung 1; diese Segmente wiederum nur zu den Teilen

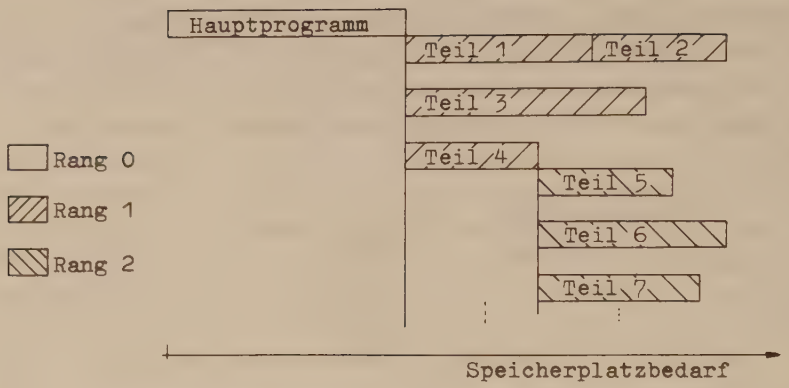


Bild 3.2.2/1 Programmüberlagerung

der Rangordnung 2, die ihnen unmittelbar zugeordnet sind. Dabei kann ein Großteil der Organisations- und Aufrufbearbeitung vom Hauptprogramm auf die Teilprogramme der Rangordnung 2 deligiert werden. Der Platz für den Programmbereich läßt sich somit besser ausnutzen, weil der Hauptteil wesentlich kürzer wird.

Aufgrund des vielschichtigen Aufbaus ist für das gesamte Programm ein besonderes Bindeprogramm notwendig, das diesen strukturellen Zusammenhang berücksichtigt. Die Ablaufsteuerung wird auch nicht vom Betriebssystem vorgenommen, sondern geschieht durch ein besonderes Steuerprogramm.

Da die Overlaytechnik die größere Vielfalt bietet, und die Struktur dem Programmlauf angepaßt werden kann, wird diese Methode für das aufzubauende Simulationsprogramm angewendet. Durch eine geschickte Aufteilung kann so nicht allein auf minimalen Arbeitsspeicherbedarf, sondern ebenfalls auf geringste Zahl von Programmwechsel und damit auf kurze Rechenzeit hingearbeitet werden.

3.3 Aufteilung des Gesamtprogramms.

Damit das Simulationsprogramm in dem verfügbaren Speicherbereich von 11 k ablaufen kann, wird es nach der zuvor beschriebenen Methode zerlegt, und zwar in ein Hauptprogramm (Mainprogram) und eine Reihe von unabhängigen Teilprogrammen (Subroutines). Die dabei notwendigen Änderungen in der Programmstruktur resultieren hauptsächlich aus dem funktionellen Ablauf des Programms. Unter Berücksichtigung der auszubauenden Dialogfunktion wird das Programm entsprechend seiner Funktion in folgende vier Gruppen eingeteilt:

1. die Abwicklung des Dialogverkehrs,
2. Einlesen, Prüfen und Umcodieren der Problembeschreibung,
3. Berechnung von Parametern und Startwerten sowie
4. die schrittweise Berechnung der Systemgrößen.

Aus der weiteren Teilung der Gruppen in funktionell abgeschlossene Programmstücke ergibt sich dann der Speicherbedarf für den Wechselbereich. Unter Einrechnung der Programmlängen für Steuerroutine, Hauptteil und FORTRAN-Bibliothek erhält man die Größe des Listenbereichs und damit den rechenbaren Problemumfang. Es ergibt sich dabei die folgende Speicheraufteilung:

6,2 %	Steuerroutine
5,3 %	Hauptprogramm
12,5 %	Wechselbereich
34,1 %	Bibliothek
41,9 %	Datenbereich

Dabei wird der verfügbare Arbeitsspeicher zu 99,6 % ausgenutzt. Diese abgeänderte Form des Simulationsprogramms ergibt eine lauffähige Version für den Prozeßrechner. Das Bild 3.3/1 zeigt die Aufteilung des Simulationsprogramms in seiner endgültigen Fassung, wobei die Funktion der Teilprogramme und ihre Länge angegeben sind.

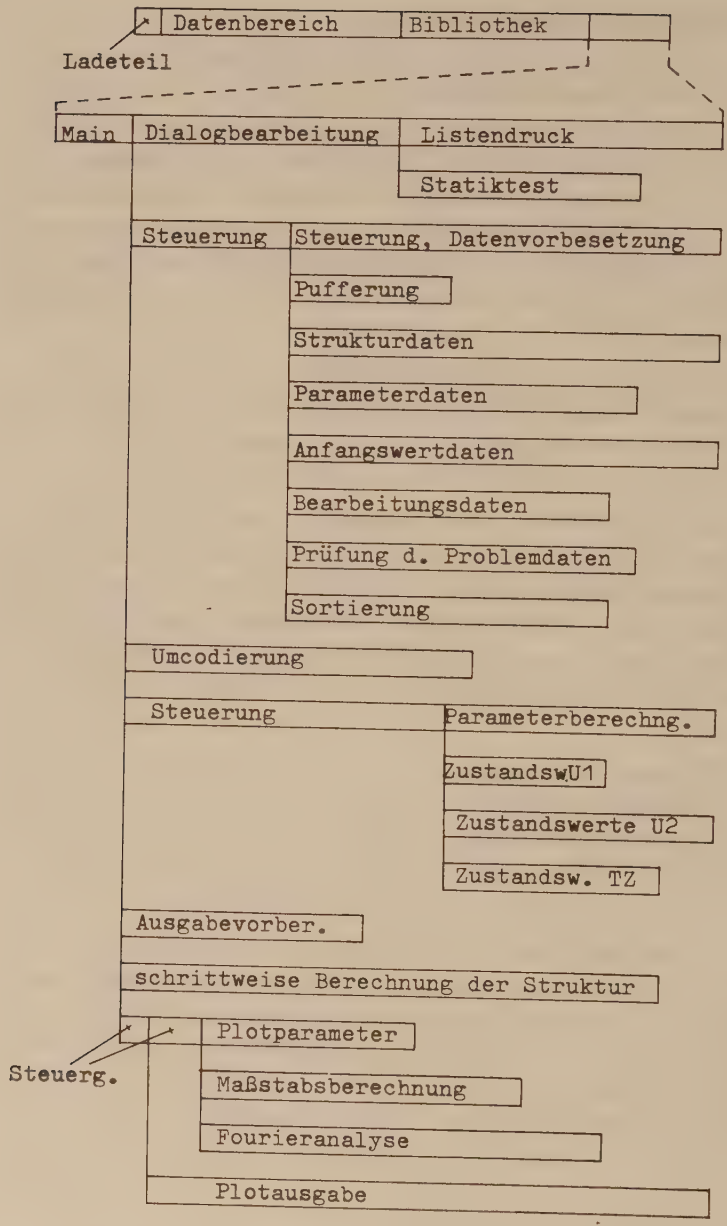


Bild 3.3/1 Gliederung des Simulationsprogramms

4 Ausbau des Simulationsprogramms.

Digitalrechenprogramme, die eine vielseitige Verwendungsmöglichkeit zulassen, erfahren von Zeit zu Zeit Verbesserungen, die sich bei häufiger und verschiedenartiger Benutzung als erforderlich herausstellen. Es ist Aufgabe der Programmwartung, die Programme daraufhin zu redigieren und Mängel oder gar Fehler zu beseitigen. Dagegen stellt man bei größeren Programmtests und andersartigen Anwendungen häufig fest, daß das Programm einer grundsätzlichen Überholung bedarf, die die inhaltliche oder strukturelle Form betrifft. Da es sich im vorliegenden Fall um grundlegende Änderungen handelt, werden sie in diesem Abschnitt eingehender erläutert.

4.1 Optimierung des Programms.

Im Hinblick auf die am Anfang gestellten Forderungen, ein Simulationsprogramm für eine kleine Rechenanlage zu erstellen und wenig Streichungen am rechenbaren Problemumfang hinzunehmen, erfüllt das Programm in der vorstehend beschriebenen, segmentierten Version diese Aufgabe. Dabei sind jedoch aus der Sicht des Benutzers zwei wesentliche Momente noch nicht berücksichtigt, die Problemgröße und der Rechenzeitbedarf.

Das Programm soll einerseits möglichst umfangreiche Problemstrukturen bearbeiten können und andererseits die verschiedensten Arten von Systemen zu beschreiben gestatten, was die Implementierung vieler Grundfunktionen bedeutet. Diese Wünsche zielen auf die Vergrößerung des gemeinsamen Datenbereiches ab. Dem kann man durch Kürzung des Programmbereiches und durch Straffung der Datenlisten entgegenkommen.

Die gewünschte kurze Programmlaufzeit ist eine Frage des funktionellen Programmaufbaus und der Qualität des FORTRAN-Kompilers.

4.1.1 Optimierung auf maximalen Problemumfang.

Eine Optimierung im Sinne einer möglichst guten Ausnutzung des gemeinsamen Datenbereiches ist schon allein deswegen angebracht, weil der FORTRAN-Kompiler nur Datenbereiche mit festen Listenlängen zuläßt, so daß unabhängig von den zu rechnenden Problemen eine Vorrausaufteilung erfolgen muß; das hat allerdings Vorteile bei der Abfassung des Rechenschemas.

Da das Simulationsprogramm nur einmal übersetzt und gebunden werden soll, und dann damit unverändert alle Probleme gerechnet werden, können die Listengrößen nicht dem jeweils anstehenden Problem angepaßt werden. Für eine optimale Auslegung des Datenbereiches müßte man aber alle noch zu rechnende Probleme kennen, denn der Umfang eines Rechenproblems bestimmt die Länge der Reihenfolgeliste, Art und Anzahl der Grundfunktionstypen und die damit verbundenen Listen. Eine Analyse bereits gerechneter Probleme liefert dabei nur grobe Anhaltswerte, weil sie nur für diese Art von Aufgabenstellungen typisch ist.

Folgende Kriterien wurden zur Festlegung für den Umfang und die Aufteilung des Datenbereiches beachtet. Zur Abdeckung eines weiten Problemkreises müssen bestimmte Grundfunktionstypen implementiert werden, worauf im Abschnitt 4.2.1 weiter eingegangen wird. Die mögliche Anzahl richtet sich nach dem Speicherbedarf pro Funktion, nach einer geschätzten Mindestzahl von erforderlichen Funktionen pro Typ und nach der Analyse gerechneter Beispiele. Da eine bestimmte Rechenaufgabe in der Regel nur einen Teil der Funktionstypen benötigt, ist die Gesamtzahl der Funktionen höher anzusetzen als die Zahl, die die Reihenfolgeliste zur Definition zuläßt. Letzlich ist der strukturelle Aufbau des gesamten Programms in der Aufteilung des Datenbereiches zu berücksichtigen.

Das Ergebnis dieser Überlegungen zeigt die Tabelle 4.1.1/1 Seite 50, die den Platzbedarf der Grundfunktionstypen und den Anteil am Datenbereich aufschlüsselt. Dazu ist der prozentuale Anteil der Grundfunktionstypen den Analyseergebnissen gegenübergestellt, wobei eine erhebliche Abweichung nur bei wenigen Funktionen auftritt.

Als weiterer Schritt wurde die Programmstruktur daraufhin ausgerichtet, daß der von den Segmentgruppen gemeinsam benutzte Datenteil minimal wird, und daß das Hauptprogramm auf die Formalaufrufe der Segmente eingeschränkt wird. Der effektive Platzgewinn gegenüber der ursprünglichen Version betrug ca. 13 % des verfügbaren Arbeitsspeichers, was eine Erweiterung des Datenbereiches um etwa 30 % bedeutet.

4.1.2 Optimierung auf minimale Laufzeit.

Da der Benutzerkreis einer kleinen Rechenanlage begrenzt ist, steht jedem einzelnen für ein Rechenvorhaben ein relativ großer Zeitraum zur Verfügung, so daß eine längere Programmlaufzeit hier nicht blockierend wirken würde. Dennoch wünscht man die Simulationsergebnisse möglichst schnell, um sich der eigentlichen Aufgabe zuwenden zu können, der Interpretation und Auswertung der Ergebnisse.

Bestimmend für die Laufzeit sind einmal programmabhängige Faktoren wie Programmstruktur, Programmteilwechsel und Übersetzungseinfluß des Kompilers, die mit der Fertigstellung des Programms festliegen; andererseits sind es die problembedingten Faktoren wie Schrittzahl sowie Typ und Anzahl der verwendeten Funktionen, die den Zeitbedarf für eine Simulation mitbestimmen.

Um die Angriffspunkte für die Zeitoptimierung zu verdeutlichen, betrachte man den Programmablauf aus Bild 2.1/2. Die Teile A, B und C werden bei der Simulation nur einmal durchlaufen, um die Problemstellung aufzubereiten. Dagegen wird der Teil D für jeden Integrationsschritt zur Funktionsberechnung benötigt. Die Laufzeit in den ersten Programmteilen zu vermindern , hätte daher wenig Erfolg, zumal der Einlesevorgang von der Geschwindigkeit des Eingabemediums bestimmt wird. Demgegenüber gehen die problemabhängigen Faktoren multiplikativ in den Zeitaufwand für den Teil D ein. Ein Zahlenbeispiel verdeutlicht diesen Zusammenhang.

Durchschnittlicher Zeitbedarf pro Funktion	2,7 ms
organisatorischer Zeitbedarf pro Schritt	0,2 ms
Zahl der zu rechnenden Funktion	100
Schrittzahl	1000

Dieses Problembeispiel würde dann im Teil D eine Rechenzeit von (100 2,7 ms + 0,2 ms) 1000 = 270 s benötigen. Der Zeitbedarf für die Programmteile A bis C kann bei Lochstreifeneingabe mit ca 15 s angesetzt werden.

Für eine zeitoptimale Auslegung des Programms muß somit der Zeitbedarf der FORTRAN-Anweisungen bestimmt werden, um gezielt in dem Programmabschnitt D einzugreifen. Die Überarbeitung bringt jedoch nur einen Zeitgewinn von etwa 20 %.

Eine erhebliche Verbesserung kann man nur dann erwarten, wenn auf die Assemblersprache zurückgegangen wird. Ausschlaggebend ist insbesondere die Adressrechnung von Feldkomponenten. Durch entsprechende Reihenfolge der Feldindizes können die Feldkomponenten teilweise über eine einfachere und damit schnellere Adressrechnung abgerufen werden. Daneben lassen sich noch einige andere FORTRAN-Anweisungen straffer fassen.

Wie aus der Gegenüberstellung in der Tabelle 4.1.1/1 hervorgeht, bewirkt die Formulierung dieses Programmteils in der Assemblersprache eine durchschnittliche Zeitverkürzung von 50 %. Als Zugabe ergibt sich eine Reduktion der Programmlänge von 10 %.

4.2 Erweiterungen des Simulationsprogramms.

Da ein Rechenproblem nur dann von dem Simulationsprogramm bearbeitet werden kann, wenn es durch die zugelassenen Vereinbarungen auszudrücken ist, schränkt der Vorrat an Grundfunktionen den rechenbaren Problemkreis ein. Ein Vergleich mit anderen Programmen und dem Analogrechner zeigt zwei Schwächen von dem ursprünglichen DIGSI auf:

a) der begrenzte Vorrat an Grundfunktionen und
b) die geringere Flexibilität gegenüber dem Analogrechner.

Die Menge der Grundfunktionen wird bei der Konzeption des neuen Programms so festgelegt, daß sich alle grundlegenden mathematischen und logischen Operationen berechnen lassen, und damit ein großer Teil der denkbaren Probleme gelöst werden kann. Die dann noch verbleibende Lücke kann zwar bei den übersetzenden Sprachen durch einen Unterprogrammanschluß geschlossen werden, bei DIGSI ist diese Möglichkeit jedoch nicht vorgesehen.

Ein Analogrechner bietet bei der Problemlösung eine höhere Benutzerfreundlichkeit, weil die Beobachtung des Simulationsablaufes und Eingriffe unmittelbar möglich sind. Dieser direkte Kontakt besteht beim Digitalrechner nicht, insbesondere bei Großrechenanlagen mit Auftragsabwicklung. Dagegen läßt sich bei kleineren Rechnern mit geeigneter Ausstattung ein Dialogsystem aufbauen, das den direkten Zugriff zur Simulationsrechnung und Ergebnisauswertung gestattet.

4.2.1 Vollständigkeit der Grundfunktionstypen.

Da bei DIGSI keine Nahtstelle vorhanden ist, die eine Definition zusätzlicher Funktionen erlaubt, muß bei diesem Programmkonzept dafür gesorgt sein, daß die Liste der Grundfunktionstypen möglichst vollständig ist. In /7/ ist eine Aufstellung der zweckmäßigen Grundfunktionstypen, dort Blockarten genannt, angeführt, wobei auch kein Anspruch auf Vollständigkeit erhoben wird. Um alle dort angegebenen Blockarten aufnehmen zu können, müßte der verfügbare Platz erheblich größer sein. Das resultiert nicht allein aus der Zahl der notwendigen Listenplätze im gemeinsamen Datenbereich, sondern folgt aus der Forderung, daß die Rechenalgorithmen aller Grundfunktionen nur in einem Teilprogramm enthalten sein sollen, damit die Simulationsrechnung zeitoptimal ablaufen kann.

Bei einem Vergleich der Grundfunktionstypen untereinander stellt man fest, daß sich einige von ihnen durch Kombination anderer ausdrücken lassen. Ein Teil mathematischer Funktionen und der Kurvengeber müssen wegen ihres Rechenaufwandes fallengelassen werden. Es wurden daher nur folgende Funktionstypen zusätzlich zur ursprünglichen Version aufgenommen:

die Zeitgeberfunktion TM,
die Abtast- und Haltefunktion TS,
die monostabile Kippstufe MF und
die logische NAND-Funktion NA.

Weitere Grundfunktionen ließen sich zwar prinzipiell noch aufnehmen, da hierfür im Quellprogramm eindeutige Schnittstellen eingeführt wurden, es müßten dann jedoch Abstriche am Datenbereich hingenommen werden.

4.2.2 Sortierung der Rechenreihenfolge.

Die Simulation beim Digitalrechner erfolgt schrittweise und sequentiell, beim Analogrechner kontinuierlich und parallel. Die Berechnung der Funktionen mit einer großen Schrittweite verursacht den Mangel an Stabilität und Kontinuität der Lösungsfunktion. Er kann zwar durch Verkleinerung der Schrittweite vermindert werden, dabei sich ergebende Rundungsfehler fördern jedoch die Instabilität. Eine andere Ursache fehlerhafter Lösungen liegt in der sequentiellen Arbeitsweise. Dadurch, daß die einzelnen Funktionen nacheinander in demselben Rechenschritt berechnet werden, entstehen bei beliebiger Reihenfolge der Funktionen zusätzliche Laufzeiten, die zu nichtstabilen oder fehlerhaften Lösungen führen.

Dieser Fehler tritt jedoch dann nicht auf, wenn die Berechnung der Funktionen in der richtigen Reihenfolge geschieht. Sie ist dann gegeben, wenn in einem Rechenschritt die Eingangswerte für die gerade zu berechnende Funktion bereits berechnet wurde, wenn sie bekannt sind. Zu den bekannten Werten zählen Konstante, Zeit- und Taktgeber sowie die Ausgänge der Funktionen, die bis dahin berechnet wurden. Als in 2. Ordnung bekannte Eingangswerte, sogenannte vorbekannte Eingänge, gelten die Ausgänge der Funktionen, die ein speicherndes Verhalten aufweisen wie Totzeitfunktion oder Übertragungsfunktionen 1. und 2. Ordnung, jedoch ohne P- oder D-Verhalten.

Die Reihenfolge der Funktionenberechnung wird durch die Eingabereihenfolge der Strukturanweisungen festgelegt. Man ent-

nimmt sie günstigerweise einem das Problem beschreibenden Blockschaltbild, in dem die Pfeile des Informationsflusses die Wirkungsrichtung und damit die richtige Reihenfolge anzeigen. Eine derartige 'manuelle' Sortierung der Funktionen ist bei größeren Problemen nicht mehr leicht zu bewältigen.

Das Simulationsprogramm wurde daher mit einer automatischen Sortierung ausgestattet, die sich an das in /7/ beschriebene Baumsortierverfahren anlehnt. Dabei wird die eingegebene Reihenfolgeliste auf Funktionen mit bekannten Eingängen abgesucht und an den Anfang gestellt. Erst in zweiter Linie wird auf vorbekannte Funktionen zurückgegriffen. Findet der Sortieroperator eine geschlossene Informationsschleife vor, in der keine speichernde Funktion enthalten ist, so wird diese als algebraische Schleife identifiziert. Eine weitere Aufschlüsselung im Sinne der Sortierung wäre dann sinnlos, der Rechenlauf wird als fehlerhaft markiert. Da die Reihenfolge der sortierten Funktionen ausgegeben werden kann, ist die algebraische Schleife erkennbar.

Auf eine automatische Beseitigung solcher Schleifen mußte verzichtet werden, weil das Grundkonzept der Simulation eine iterative Berechnungsweise nicht zuläßt. Das willkürliche Einfügen eines Schrittspeichers beseitigt zwar formal den Fehler, jedoch kann man die Auswirkungen auf die übrige Struktur nicht voraussagen. Es bleibt dem Anwender überlassen, das Problem zu überarbeiten oder auf eine weitere Sortierung zu verzichten und den Fehler durch Verkleinerung der Schrittweite gering zu halten.

Eine automatisch sortierte Struktur enthält keine zusätzlichen Laufzeiten, die Lösungen weisen nur noch schrittweitenbedingte Fehler auf.

4.2.3 Plotausgabe und Fourieranalyse.

Die Simulation bringt als Ergebnis eine gewünschte Menge von Daten hervor, deren Interpretation dem Anwender überlassen ist. Ein erweitertes Simulationsprogramm kann dabei wertvolle Hilfsdienste leisten.

Für eine maschinelle Auswertung der Ergebnisdaten sind diese auf einem Massenspeicher hinterlegt. Da ein Großteil von Simulationsläufen als vorläufig betrachtet werden kann, weil sich aufgrund der Auswertung der Ergebnisse Änderungen der Problemdaten ergeben, ist die Auswertung mit gesonderten Programmen nicht effektiv; es müßte dazu jeweils ein Programmwechsel erfolgen, wobei folglich für jeden neuen Simulationslauf alle Problemdaten übergeben werden müssen. Deshalb wurde in dem Simulationsprogramm eine Nahtstelle geschaffen, an der Auswerteprogrammteile direkt angebunden werden können, und somit ein Programmwechsel vermieden wird. Es bestehen nunmehr zwei häufig benötigte Auswertungshilfen.

Eine graphische Darstellung der Ergebnisse liefert dem Anwender ein wesentlich anschaulicheres Bild als eine Tabelle. Die Ausstattung der Rechenanlage S 305 mit einem graphischen Sichtgerät kommt diesem Wunsch entgegen. An die Nahtstelle wurde daher ein Teilprogramm zur Erstellung eines Plots angeschlossen, das Folgendes leistet. Die Daten werden als stetige Kurven in Abhängigkeit von der unabhängigen Variablen mit dem zugehörigen Achsensystem aufzeichnet. Die Kurven können durch unterschiedliche Strichlierung voneinander unterschieden werden. Die Maßstäbe für die vorgegebene Schirmbildgröße lassen sich automatisch bestimmen. Ausschnittvergrößerungen sind möglich.

Als weitere Auswertung ist die Bestimmung der Fourierkoeffizienten periodischer Ergebnisfunktionen zweckmäßig. Diese Analyse läßt sich deshalb einfach durchführen, weil hier das Integrationsverfahren mit konstanter Schrittweite arbeitet, so daß die Stützpunkte für die diskrete Fourieranalyse bereits äquidistant vorliegen und nicht durch eine Interpolation bestimmt werden müssen.

Mit Hilfe des endlichen Fourierreihenansatzes läßt sich eine stetige Funktion f(x) bestimmen, die mit den Funktionswerten in den 2N Stützpunkten übereinstimmt.

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{N-1} \left(a_k \cos(\tfrac{\pi}{N} k x) + b_k \sin(\tfrac{\pi}{N} k x) \right) + \frac{a_N}{2} \cos(\pi x) \qquad (4.2.3/1)$$

Je höher die Zahl der Abtastpunkte ist, um so mehr gleicht die Reihe der wahren, gesuchten Funktion. Die Koeffizienten a_k und b_k werden durch die Summationen

$$a_k = \frac{1}{N} \sum_{x=0}^{2N-1} f^*(x) \cdot \cos(\tfrac{\pi}{N} k x) \qquad k = 0,1,2, \ldots N$$

$$b_k = \frac{1}{N} \sum_{x=0}^{2N-1} f^*(x) \cdot \sin(\tfrac{\pi}{N} k x) \qquad k = 1,2, \ldots N \qquad (4.2.3/2)$$

bestimmt, wobei $f^+(x)$ die abgetastete Funktion ist. Da die beiden Winkelfunktionen insgesamt je (2N·k)-mal berechnet werden, setzt man besser dafür an den 2N Stützpunkten eine Rekursionsformel ein /8/.

$$\begin{aligned} \sin(x_i) &= \sin(x_0) \cdot V_{i-1} & \sin(x_0) &= \sin(\tfrac{\pi}{N} k) \\ \cos(x_i) &= \cos(x_0) \cdot V_{i-1} - W_{i-1} & \cos(x_0) &= \cos(\tfrac{\pi}{N} k) \\ V_i &= 2 \cdot \cos(x_0) \cdot V_{i-1} - W_{i-1} & V_0 &= 1 \\ W_i &= V_{i-1} & W_0 &= 0 \end{aligned} \qquad (4.2.3/3)$$

Die Winkelfunktionen werden so nur noch für k Koeffizientenpaare berechnet, wodurch Rechenzeit eingespart wird. Der zusätzliche Rechenfehler liegt bei sechsstelliger Mantisse und bei 500 Abtastpunkten pro Periode unter 0,01%.

4.2.4 Dialogbearbeitung und Statiktest.

Zu den entscheidenden Vorteilen des Analogrechners zählen neben der hohen Rechengeschwindigkeit die enge Verbindung zwischen Anwender und Simulator sowie die Möglichkeit der schnellen Fehlersuche durch den statischen Test. Dieser Vorsprung konnte durch den Ausbau von DIGSI zu einem Dialogsystem eingeholt werden.

Der Simulationsablauf kann an jeder sinnvollen Stelle unterbrochen werden, um Änderungen an der Problembeschreibung vorzunehmen, den Simulationsablauf zu beeinflussen oder um über den Zustand der Simulation Informationen abzurufen. Die Unterbrechung kann von außen durch einen Konsolschalter initiiert werden, der nach den Programmabschnitten A bis D (Bild 2.1/2) auf seinen Zustand abgefragt wird.

Die Fehlerdiagnose ist außer der Erkennung algebraischer Schleifen durch weitere Formaluntersuchungen komplettiert worden. Die Problembeschreibung wird danach vollständig auf richtige Funktionsdefinition, Eingangsverknüpfungen und Parameterzuweisung untersucht. Da nunmehr die gesamte Beschreibung zeilenweise korrigiert werden kann, läßt sich eine fehlerfreie Problembeschreibung immer leicht herstellen.

Das führt zwar zu einer ablauffähigen Simulation, schließt jedoch problemlogische Fehler nicht aus, zum Beispiel die Vertauschung von Eingängen; sie aufzudecken, erfordert die Mitarbeit des Anwenders. Beim Analogrechner können mit Hilfe der Testfunktion 'Statischer Test' und 'Halt' solche Fehler schnell erkannt werden. Eine gleiche Testfunktion wurde daher in das Simulationsprogramm aufgenommen, mit der bei jeder Unterbrechungsmöglichkeit eine Werteliste aller Funktionen erstellt wird. Dem statischen Test entspricht die Wertetabelle nach der Startwertberechnung und die Unterbrechung nach jeweils einem Rechenschritt der Funktion Halt. Damit sind dem Anwender Hilfsmittel an die Hand gegeben, wie sie sonst nur der Analogrechner bietet.

4.2.5 Automatische Simulation mit Parametervariation.

In der aufgezeigten Konzeption wird eine Simulationsaufgabe in der Form eines Dialoges abgewickelt. Das ist für die Mehrzahl der Probleme zweckmäßig, bei denen das dynamische Verhalten in einem bestimmten Zustand interessiert. Die Bearbeitung führt selbst dann noch zu annehmbaren Laufzeiten, wenn die Zeitkonstanten des Problems sehr unterschiedlicher Grössenordnung sind; durch die Wahl verschiedener Schrittweiten erreicht man eine kürzere Problembearbeitungszeit. Allerdings wird die Laufzeit nur dann wesentlich gesenkt, wenn mehr als 50 % der Funktionen mit anderer Schrittweite gerechnet werden.

Bei Aufgaben, bei denen man sich für die stationären Endwerte der Systemgrößen innerhalb ausgedehnter Parameterbereiche interessiert, ist die Dialogform des Simulationsablaufes nicht praktikabel. Die Ergebnisse eines einzigen Betriebspunktes stehen erst nach Ablauf der Einschwingvorgänge, das heißt nach längerer Zeit zur Verfügung. Es wurde deshalb eine zweite Version zur automatischen Bearbeitung von Simulationsaufträgen entwickelt. In diesem Falle werden die einzustellenden Parameter zuvor zwischengespeichert. Das Problem kann im Dialog getestet und dann an beliebiger Stelle auf die automatische Arbeitsform umgesteuert werden. Alle im Dialog ansprechbaren Programmfunktionen werden dann unterdrückt. Bei Erreichen einer vorgegebenen Genauigkeit bezüglich der Konstanz der Systemgrößen wird das Ergebnis gespeichert und ein neuer Simulationslauf mit dem nächsten Satz von Parametern automatisch gestartet.

Funktionstyp	Platzbedarf d. Funktion			prozent. Anteil d. Funktionstyps		Zeitbedarf zur Berechnung eines Funktionswertes [ms]	
	Datenworte	implementiert Anzahl	Anteil im Datenbereich [%]	der impl. Funktion	gemäß d. Analyse	FORTRAN	Assembler
AD	4	70	5,6	11,4	11,0	2,65	1,60
SU	2	70	2,8	11,4	12,3	2,35	1,25
UO	4	70	5,6	11,4	9,0	2,30	1,20
U1	11	30	6,6	4,9	10,0	3,10	1,60
U2	21	20	8,4	3,3	3,3	4,85	2,20
TZ	29	10	5,8	1,6	1,0	2,85	1,30
EK	2	40	1,6	6,5	11,0	-	-
WU	2	10	0,4	1,6	0,2	2,85	1,70
MO	2	10	0,4	1,6	1,5	2,90	1,75
SI	2	10	0,4	1,6	1,7	3,50	2,35
AB	2	15	0,6	2,5	1,0	2,40	1,30
IN	2	10	0,4	1,6	-	2,50	1,40
MU	2	20	0,8	3,3	4,6	2,35	1,25
DI	2	20	0,8	3,3	1,3	2,95	1,90
FG	62	5	6,2	0,8	1,0	3,30	1,85
BG	6	15	1,8	2,5	0,9	2,50	1,30
SW	4	25	2,0	4,0	6,9	2,65	1,60
TS	2	15	0,6	2,5	-	2,40	1,30
KO	1	30	0,6	4,9	8,6	2,30	1,20
FF	5	25	2,5	4,0	3,6	2,55	1,40
MF	6	5	0,6	0,8	0,2	2,50	1,40
UN	1	30	0,6	4,9	4,2	2,15	1,10
OR	1	30	0,6	4,9	3,8	2,15	1,10
NI	1	15	0,3	2,5	2,3	2,15	1,10
NO	1	8	0,16	1,3	-	2,15	1,10
NA	1	7	0,14	1,1	0,7	2,15	1,10
Mittelw.	6,85	23,7	2,16			2,66	1,45
Gesamt		615	56,3				
Struktur	8	250	39,8				
Sonstiges			3,2				
Reserve			0,7				

Tabelle 4.1.1/1 Speicher- und Zeitbedarf der Funktionen

5 Die digitale Simulation von Netzwerken mit Schalterelementen.

Nichtlineare Netzwerke, zu denen auch Stromrichterschaltungen gehören, sind einer geschlossenen, analytischen Lösung schwer zugänglich. Deshalb werden diese Schaltungen mit Analog- oder Digitalrechnern nachgebildet. Bei der digitalen Simulation sind dafür spezielle Netzwerkprogramme bekannt /14,15/, die auf Stromrichterschaltungen zugeschnitten sind; ihre Funktion wird im nachfolgenden Abschnitt kurz erläutert.
Auf der anderen Seite kann man das dem Netzwerk zugehörige Gleichungssystem mit dem Analogrechner oder mit gleichwertigen digitalen Simulationsprogrammen lösen. Für die Nachbildung auf dem Analogrechner werden in /3/ Lösungen angegeben, die sich auf eine spezielle Klasse von Stromrichterschaltungen beziehen. Die Behandlung von derartigen Schaltungen mit Digitalprogrammen wird in den anschließenden Abschnitten angeführt und an einigen Beispielen erläutert.

5.1 Die Simulation von Stromrichterschaltungen durch Netzwerkprogramme.

Bei diesen Programmen werden die Maschenumlaufgleichungen $\sum u_{vm} = 0$ in Verbindung mit den Knotengleichungen $\sum i_v = 0$ zur Bestimmung der Zustandsgrößen i_{vm} der unabhängigen Maschen für jeden Schritt Δt gelöst. Die Parameter der Schaltung (Impedanzen, Quellen, Ventile) werden über eine topologische Beschreibung des Netzwerkes übermittelt. Die dabei notwendige Zweig-Maschen-Inzidenzmatrix muß bei /15/ vom Benutzer selbst aufgestellt werden. Die Berechnungsweise erfolgt in Matrizenform.

Im Hinblick auf Stromrichterschaltungen ist die Nachbildung von den schaltenden Elementen unterschiedlich gelöst. In /15/ ist der Ersatz durch eine R-L-Kombination gewählt, bei /14/ werden gesteuerte Strom- beziehungsweise Spannungsquellen eingesetzt. Das Gleichungssystem ist somit abschnittsweise bis zu einer Änderung eines Schaltzustandes konstant; danach müssen die entsprechenden Koeffizientenmatrizen geändert werden.

Mit Hilfe dieser Programme lassen sich bei den Stromrichterschaltungen die Funktionsfähigkeit und einzelne Betriebszustände bestimmen. Die Ergebnisse liegen dabei als Zeitverläufe gewünschter Strom- und Spannungswerte vor. Eine Auswertungserleichterung bietet /14/ durch die harmonische Analyse.

Diese Simulationsmethode erfordert als Eingabedaten die Schaltungsparameter, die topologische Beschreibung sowie Angaben über die Zündwinkel und liefert als Ergebnis Zeitverläufe der Zustandsgrößen. Zusätzliche Ausgabewerte (Leistungsgrößen, Effektivwerte usw.) sind nicht möglich, so daß auch somit keine neuen Eingangsgrößen gewonnen werden können. Verknüpfungsrechnungen außerhalb der reinen Nachbildung der Schaltung können nicht gleichzeitig vom Programm bearbeitet werden. Sofern derartige Probleme vorliegen, muß auf allgemeine Simulationsprogramme ausgewichen werden.

5.2 Die Simulation von Stromrichterschaltungen mit allgemeinen Simulationsprogrammen.

Allgemeine Simulationsprogramme setzen ebenso wie der Analogrechner für die Problembearbeitung ein mathematisches Modell voraus, das mit den implementierten Rechenelementen verarbeitet werden kann. Bei elektrischen Netzwerken besteht das Modell aufgrund induktiver oder kapazitiver Schaltungselemente aus einem Satz von Integro-Differentialgleichungen. Sie bestimmen sich aus den Maschen- und Knotengleichungen des Netzwerkes. Weil differenzierende Übertragungsfunktionen nicht implementiert sind, müssen die Gleichungen nach den Ableitungen der Zustandsgrößen aufgelöst werden, die bei Stromrichterschaltungen wegen der hauptsächlich induktiven Schaltungselemente die Maschenströme sind. Sofern es sich um ein lineares Netzwerk handelt, ist das Gleichungssystem in dieser Form integrierbar.

Stromrichterschaltungen enthalten jedoch Ventile als nichtlineare Elemente, so daß das Gleichungssystem wegen der nicht konstanten Koeffizienten nicht für den gesamten Zeitbereich integrierbar ist. Wie im Abschnitt 5.1 angegeben, werden bei

den Netzwerkprogrammen die Ventile durch lineare Schaltungselemente ersetzt, wodurch das Gleichungssystem zumindest abschnittsweise linear ist. Diese Darstellungsweise läßt sich für das Programm DIGSI nicht verwenden, weil jede Änderung eines Ventilzustandes die Eingabe von neuen Parametern erfordern würde; ein hoher organisatorischer Aufwand wäre dabei erforderlich.

Eine abschnittweise Modelldarstellung, wie sie an den Beispielen in /7/ angeführt ist, bedingt je nach der Zahl der möglichen Schaltzustände eine vielfache Anzahl von Gleichungen, die parallel dargestellt und gelöst werden müssen, wobei nur ein Teil für die gewünschte Lösungsfunktion wesentlich ist.

In /3/ wird ein Modell für Ventile angegeben, das die Dioden oder Thyristoren über eine Schaltfunktion durch hoch- oder niederohmige Widerstände ersetzt. Diese Darstellung ist zwar für den Analogrechner geeignet, führt jedoch bei der digitalen Simulation zu sehr kleinen Rechenschrittweiten, um Instabilitäten der Lösung zu vermeiden.

Da die Simulationsprogramme eine Reihe von logischen Funktionen enthalten, bietet es sich an, den Schaltzustand der Ventile damit auszudrücken. Hierfür sind zusätzliche Gleichungen zur Bestimmung der Ventilspannung u_T und des Ventilstromes i_T erforderlich, die aus den zugehörigen Maschen und Knoten ermittelt werden können. Ein derart abgeleitetes Gleichungssystem ist für die digitale Simulation oft zu aufwendig.

a) Um aus den Netzwerkgleichungen die integrierbare Form sowie die Gleichungen für die Ventilgrößen zu erhalten, ist ein erheblicher mathematischer Aufwand erforderlich.
b) Die Gleichungskoeffizienten enthalten eine Reihe von Schaltungsparametern, die bei mehreren Koeffizienten auftreten. Wird ein einziger Parameter geändert, muß die Mehrzahl der Koeffizienten neu bestimmt werden.
c) Das Modell enthält als Variable die Zustandsgrößen. Alle anderen für die Simulation wichtigen Größen müssen erst über zusätzliche Gleichungen ermittelt werden.

Das Modell wäre wesentlich günstiger, wenn die Gleichungen so angegeben werden könnten, daß sie unmittelbar in integrierbarer Form vorliegen, und die Schaltungsparameter nur einmal als Koeffizienten auftreten. Solche direkt integrierbare Gleichungen erhält man dann, wenn die Zustandsgröße eines Zweiges ebenfalls die einer Masche ist (Zweigstrom = Maschenstrom). Das läßt sich immer damit erreichen, daß die gegebene Schaltung durch geeignete, hochohmige Hilfszweige ergänzt wird, wobei die Zweige sich hierüber zu einer Masche schliessen. Die entsprechenden Maschenumlaufgleichungen liegen dann sofort in der integrierbaren Form vor. Dadurch wird das Netzwerk für die Simulation zunächst entkoppelt; die Verknüpfung der Gleichungen geschieht über die Hilfszweige.

5.2.1 Das Modell einer n-pulsigen Mittelpunktschaltung.

Das Schaltbild einer Stromrichtermittelpunktschaltung ist in Bild 5.2.1/1 dargestellt. Das Netzwerk enthält n+1 Zweige und ist durch einen parallelen Hilfszweig mit dem Widerstand R ergänzt.

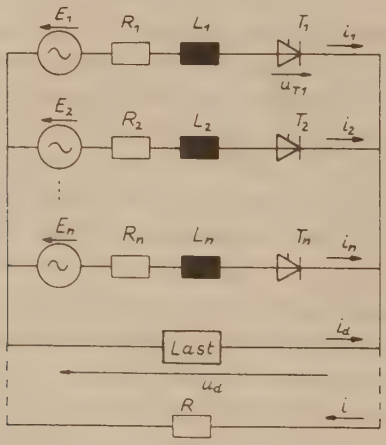


Bild 5.2.1/1 Stromrichtermittelpunktschaltung

Obwohl der Lastzweig kein Ventil enthält, kann dieser zunächst formal gleich den anderen behandelt werden, wobei danach eine

Korrektur anzubringen ist. Das gleiche gilt für Zweige ohne Quellspannungen.
Die Maschenumlaufgleichung des Zweiges k über den Hilfszweig lautet:

$$-E_k + L_k \cdot \frac{d}{dt}(i_k) + R_k \cdot i_k + u_{Tk} + u_d = 0 \qquad (5.2.1/1)$$

Die Koeffizienten der Gleichung sind Schaltungsparameter und treten in den übrigen n Gleichungen nicht auf. Die Gleichung ist integrierbar und kann damit unmittelbar als Modellgleichung herangezogen werden.
Zur Ableitung der Ventilgrößen wird definiert:

$$V_k = -E_k + R_k \cdot i_k + u_d \qquad (5.2.1/2)$$

und so ergibt sich die Gl. (5.2.1/1) zu:

$$L_k \cdot \frac{d}{dt}(i_k) = -V_k - u_{Tk} \qquad (5.2.1/3)$$

Die Einführung einer Schaltfunktion S_k für das Ventil k, die den Zustand 'leitend' mit $S_k = 1$ und 'gesperrt' mit $S_k = 0$ angibt, spaltet die Maschengleichung in zwei Gleichungen auf. Dabei ergibt sich eine Bestimmungsgleichung für die Ventilspannung; der Ventilstrom ist immer gleich dem Zweigstrom.

$$\begin{array}{lll} S_k = 0 & u_{Tk} = -V_k & \frac{d}{dt}(i_k) = i_k = 0 \\ S_k = 1 & u_{Tk} = 0 & L_k \cdot \frac{d}{dt}(i_k) = -V_k \end{array} \qquad (5.2.1/4)$$

Die Spannung u_d läßt sich über den Hilfszweig bestimmen.

$$u_d = R \cdot \sum_{k=1}^{n+1} i_k \qquad (5.2.1/5)$$

Für die Simulation ergibt sich daraus das Blockschaltbild für den Zweig k nach Bild 5.2.1/2. Es enthält neben der Integration der Gl. (5.2.1/3) die Bestimmung der Schaltfunktion.

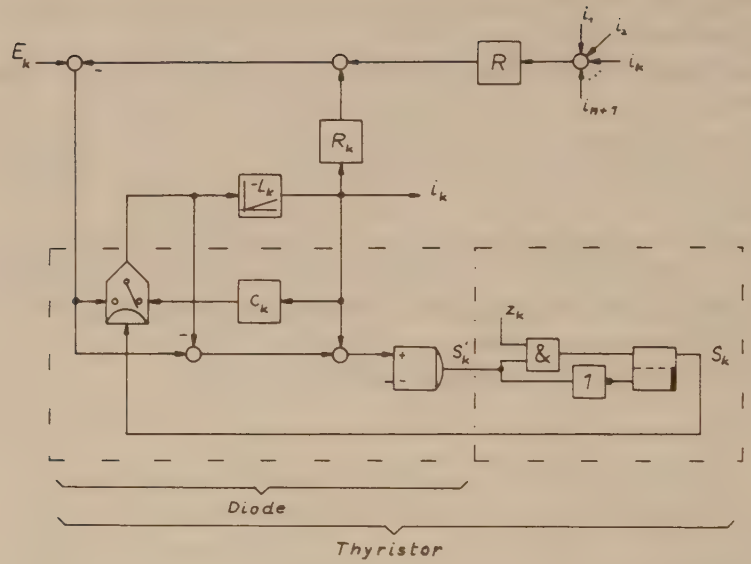


Bild 5.2.1/2 Blockschaltbild des Zweiges k

S_k ist gleich 1, wenn die Ventilspannung positiv und bei Thyristoren die Zündimpulsfunktion $z_k = 1$ ist, oder wenn im leitenden Zustand der Ventilstrom positiv ist. Die Ventilspannung nach Gl. (5.2.1/4) kann gemeinsam mit dem Zweigstrom i_k auf den Komparator geschaltet werden, der die Funktion S_k' erzeugt, weil die Ventilspannung Null ist, wenn ein Zweigstrom fließt, und umgekehrt. Bei Dioden ist $S_k = S_k'$, bei Thyristoren wird die Schaltfunktion S_k mit Hilfe der Zündimpulsfunktion z_k über die Gatter und dem nachgeschalteten Flipflop hergestellt.

Negativer Zweigstrom setzt die Schaltfunktion S_k' bzw. S_k in den Zustand Null zurück. Der Abbau des Rückstromes erfolgt am Integrator über die Schleife mit den Faktor c_k. Dabei muß bei einer Rechenschrittweite T1 die Verstärkung nach der Beziehung

$$c_k < L/T1$$

gewählt werden, um den Zweigstrom in endlich kurzer Zeit jedoch ohne weitere Vorzeichenwechsel auf Null zu erzwingen.

Für Zweige, die nicht alle Schaltungsparameter enthalten, wie sie in der Gl. (5.2.1/1) aufgeführt sind, entfallen die entsprechenden Elemente in dem allgemeinen Blockschaltbild des Zweiges k.

Der hinzugefügte Hilfszweig in der Stromrichterschaltung kann auch als Regler interpretiert werden, der eine Spannungsquelle u_d so einstellt, daß die Stromsumme $i = \sum i_k$ zu Null wird. Wird der Hilfszweig durch einen hochohmigen Widerstand R gebildet, so entspricht dies einer Spannungsregelung mit Proportionalregler; der stationäre Fehler des Stromes i hängt dabei von der Verstärkung beziehungsweise von der Größe des Widerstandes ab. Bei vorgegebener Verstärkung kann die Integrationsschrittweite höchstens gleich der kleinsten auftretenden Zeitkonstante gewählt werden $T1 \leq T_{kmin}$. Dadurch ergibt sich eine mit der geforderten Genauigkeit steigende hohe Schrittzahl für die Simulation, wie es in dem Beispiel in 5.3.1 gezeigt wird.

Erhält der Regler, der auf die Spannungsquelle wirkt, PI-Verhalten, das entspricht einem Hilfszweig mit einem R-C-Reihenglied, kann die Schrittzahl gesenkt werden, da hierbei der stationäre Fehler verschwindet. Die Schrittweite ist bei vorgegebener Verstärkung nach der kleinsten Eigenzeit auszulegen. Die Nachstellzeit des Reglers richtet sich dann nach der zweitgrößten Zeitkonstanten, um ein übermäßiges Einschwingen zu vermeiden.

Enthält die Mittelpunktschaltung einen Freilaufzweig, der eine vergleichsweise kleine Impedanz enthält, so kann dieser Zweig mit dem Hilfszweig zusammengefaßt werden, wenn der Innenwiderstand vernachlässigt wird. Der Strom dieses Zweiges l und die Ventilspannung bestimmt sich mit der Schaltfunktion S_l zu:

$$\left.\begin{array}{lll} S_l = 0 \quad i_l = 0 & & u_{Tl} = -u_d \\ S_l = 1 \quad i_l = i - \sum i_k \; ; \; k \neq l & & u_{Tl} = 0 \end{array}\right\} = -\left(i \cdot R + \frac{1}{C}\int i\,dt\right) \qquad (5.2.1/6)$$

Weil durch die Vernachlässigung der kleinen Impedanz des Frei-

laufzweiges eine kleine Zeitkonstante entfällt, kann die Schrittweite vergrößert werden. Die Nachbildung dieses Zweiges reduziert sich auf die Bestimmung der Schaltfunktion S_1.

5.2.2 Stromrichterschaltungen mit parallelen Ventilzweigen innerhalb einer Kommutierungsgruppe.

Bei der Ableitung des Modells für die Mittelpunktschaltung wurde gezeigt, daß zur Bestimmung des Schaltzustandes eines Ventils in dem zugehörigen Zweig Impedanzen liegen müssen; lediglich ein einziger, reiner Ventilzweig darf vorhanden sein, wenn die Schaltung um einen Hilfszweig ergänzt wurde. Dem abgeleiteten Modell ähnlich sind Stromrichterschaltungen, bei denen innerhalb einer Kommutierungsgruppe einem Ventil weitere, induktivitätsfreie Ventilzweige parallelgeschaltet sind, wie es beispielsweise bei den zwangsgelöschten Stromrichtern der Fall ist. Um auch hier integrierbare Gleichungen zu erhalten, müßten zusätzliche Induktivitäten eingeführt werden, die vergleichsweise klein anzusetzen sind, damit die Abweichungen gegenüber der realen Schaltung gering bleiben. Das würde jedoch wegen der zwangsläufig kleinen Rechenschrittweite zu längeren Rechenzeiten führen, ohne daß damit ein wesentlicher Informationsgewinn verbunden wäre.

Enthalten die parallelen Ventilzweige keine Innenwiderstände oder nur Kondensatoren, so wird der Strom überlappungsfrei zwischen den Ventilen geschaltet. Der Zweigstrom wird dabei nicht unterbrochen, so daß die Umschaltung keine Wirkung nach außen hat. Ein solcher Zweig kann damit wie die übrigen Zweige im Modell dargestellt werden, wenn eine Verriegelung der parallelen Ventile für ihre Schaltfunktionen vorgesehen wird. Für die Zünd- und Löschbedingungen der beteiligten Ventile müssen zusätzliche Gleichungen aufgestellt werden. Ein Beispiel für eine derartige Schaltung ist der Gleichstromsteller, der im Abschnitt 5.3.2 näher erläutert wird.

5.2.3 Stromrichterschaltungen mit widerstandslosen Maschen.

Zur Vermeidung kleiner Rechenschrittweiten ist es zweckmäßig, kleine Impedanzen in den Zweigen fortzulassen, die allein der Bestimmung integrierbarer Gleichungen dienen, und nur die für

die Funktion der Schaltung wichtigen Ersatzgrößen beizubehalten. Mit dieser reduzierten Form treten in der Schaltung Betriebszustände auf, bei denen mehrere Ventile in dem Netzwerk sich zu einer widerstandslosen Masche schließen. Das ist der Fall bei Brückenschaltungen mit unsymmetrischer Steuerung bei kleinen Aussteuerungen und mit symmetrischer Steuerung im Kommutierungsfall. Die Ventilströme wie in Bild 5.2.3/1 sind dann nicht definiert, denn es kann ein endlich großer Kreisstrom i_0 fließen, ohne daß dies einen Einfluß auf den Lastkreis hat.

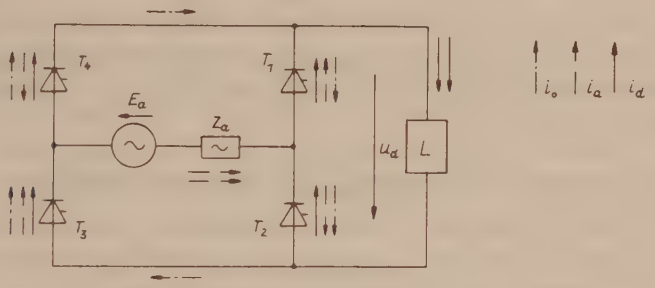


Bild 5.2.3/1 Brückenschaltung ohne Impedanz in den Ventilzweigen.

Ohne eine zusätzliche Bedingung in der Simulation lassen sich die Zweigströme nicht bestimmen. In der Praxis sorgen die immer vorhandenen Leitungsinduktivitäten für eine annähernd gleiche Aufteilung des vom Netz gelieferten Kommutierungsstromes auf die Ventilgruppen. Definierte Verhältnisse lassen sich nur dadurch schaffen, daß eine völlig symmetrische Steuerung ausgeschlossen wird, Das kann man damit erreichen, daß die Zündimpulse gegenseitig verriegelt werden; ein Folgeventil wird erst dann gezündet, wenn die Überlappung bereits abgeschlossen ist. Die Kommutierungsgruppen kommutieren dann höchstens unmittelbar nacheinander.

Eine weitere Möglichkeit ergibt sich durch Definition der Stromverhältnisse. Aus den eingezeichneten Maschenströmen berechnen sich die Ventilströme zu:

$$i_1 = i_d + i_a - i_o \quad ; \quad i_2 = i_d - i_a - i_o$$
$$i_3 = i_d + i_a + i_o \quad ; \quad i_4 = i_d - i_a + i_o \qquad (5.2.3/1)$$

Definiert man den Kreisstrom i_0 zu $i_0 = i_d - /i_a/$, so verhindert man damit, daß die Ventilströme i_1 und i_2 gleichzeitig ungleich Null sein können.

$$
\begin{aligned}
i_1 &= i_a + /i_a/ \\
i_2 &= -i_a + /i_a/ \\
i_3 &= i_a - /i_a/ + 2\,i_d \\
i_4 &= -i_a - /i_a/ + 2\,i_d
\end{aligned}
\qquad (5.2.3/2)
$$

Damit sind alle vier Ventilströme auch dann eindeutig definiert, wenn alle Ventile gleichzeitig gezündet würden. Diese arithmetische Lösung bewirkt dabei das Gleiche wie die Zündungsverriegelung.

5.3 Simulationsbeispiele von Mittelpunktschaltungen.

Anhand zweier Beispiele, einer Einpulsschaltung und eines Gleichstromstellers, wird in diesem Abschnitt die Simulation von Mittelpunktschaltungen dargestellt. Dabei wird verdeutlicht, wie aus dem allgemeinen Gleichungssystem die Zweige sich schematisch darstellen lassen. Die verschiedene Auslegung des Hilfszweiges und die Wirkung als Proportional- sowie als Proportional-Integral-Regler wird erläutert. Sein Einfluß auf die Genauigkeit und auf die Schrittweite wird hervorgehoben.

Als Beispiel für eine Schaltung, in der ein Zweig parallele Ventilzweige enthält, wird eine einfache Gleichstromstellerschaltung simuliert.

5.3.1 Einpulsstromrichter mit Nullanode.

Das Bild 5.3.1/1 zeigt die Schaltung eines Einpulsstromrichters mit Freilaufdiode, der auf einem Lastkreis mit Gegenspannung arbeitet.

Wenn bei der Mittelpunktschaltung mit direkt integrierbaren Gleichungen gearbeitet werden soll, muß die Stromaufteilung durch einen zusätzlichen Hilfszweig geregelt werden, wie er im Schaltbild als zusätzlicher Widerstand beziehungsweise als R-C-Glied auftritt.

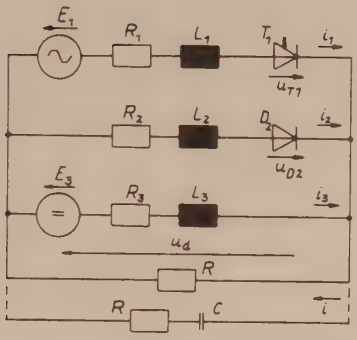


Bild 5.3.1/1 Einpulsstromrichter mit Nullanode

Wenn jedoch der Innenwiderstand des Freilaufzweiges vergleichsweise sehr klein ist, kann der Innenwiderstand auch völlig vernachlässigt werden. Beide Varianten werden ausgeführt.

a) Freilaufzweig mit Innenwiderstand.
Aus den allgemeinen Modellgleichungen für die Mittelpunktschaltung können die Gleichungen für die vorliegende Schaltung leicht hergeleitet werden.

$$V_1 = -E_1 + R_1 \cdot i_1 + u_d \quad ; \quad L_1 \cdot \frac{d}{dt}(i_1) = -V_1 - u_{T1}$$

$$V_2 = R_2 \cdot i_2 + u_d \quad ; \quad L_2 \cdot \frac{d}{dt}(i_2) = -V_2 - u_{D2}$$

$$V_3 = -E_3 + R_3 \cdot i_3 + u_d \quad ; \quad L_3 \cdot \frac{d}{dt}(i_3) = -V_3$$

$$R \cdot \sum_{k=1}^{3} i_k = u_d$$

Daraus ergibt sich das Blockschaltbild nach Bild 5.3.1/2a, in dem außerdem noch die Quellspannung E_1 und die Zündimpulsfunktion z_1 enthalten sind. Die zugehörige Beschreibung in der Sprache DIGSI enthält die Auflistung in Bild 5.3.1/3.

b) Freilaufzweig ohne Innenwiderstand.
In dem Schaltbild 5.3.1/1 werden die Impedanzen im Zweig 2 gestrichen $L_2 = R_2 = 0$. Damit ergeben sich mit den allgemeinen Modellgleichungen und der Vorschrift nach Gl. (5.2.1/6) die folgenden Schaltungsgleichungen.

$$V_1 = -E_1 + R_1 \cdot i_1 + u_d \quad ; \quad L_1 \cdot \frac{d}{dt}(i_1) = -V_1 - u_{T1}$$

$$V_2 = u_d \quad ; \quad i_2 = i - \sum_{k=1}^{3} i_k \qquad k \neq 2$$

$$V_3 = -E_3 + R_3 \cdot i_3 + u_d \quad ; \quad L_3 \cdot \frac{d}{dt}(i_3) = -V_3$$

$$R \sum_{k=1}^{3} i_k = u_d$$

Das Blockschaltbild ändert sich gegenüber dem Fall a) in der Darstellung des Zweiges 2 und in der Bestimmung der Gleichspannung u_d. Das Bild 5.3.1/2b ist hierfür gültig.

Für beide Fälle wurde die Simulation mit proportional und mit proportional-integral wirkender u_d- Regelung durchgeführt. Die Bilder 5.3.1/4 und 5.3.1/5 zeigen die Spannungs- und die Stromverläufe für einen stationären Betriebsfall bei einem Anschnittwinkel von $\alpha = 80^\circ$. Beim Proportionalregler ist wie erwartet in beiden Fällen eine Regelabweichung im Verlauf des Gleichstromes gegenüber dem Netzstrom vorhanden, die jedoch durch den integralen Anteil beseitigt werden kann. Außerdem kann bei einem PI-Regler mit einer größeren Schrittweite gerechnet werden, ohne daß die Genaugkeit der Simulation vermindert wird.

Bild 5.3.1/6 zeigt für einen anderen Betriebspunkt (lückender Laststrom) den Verlauf der Gleichspannung u_d bei verschiedenen Einstellungen der Verstärkung (R), der Integrationskonstanten (T = RC) und der Schrittweite (T1). Der Einfluß der eingestellten Variationen um den Faktor 2 kann als gering betrachtet werden.

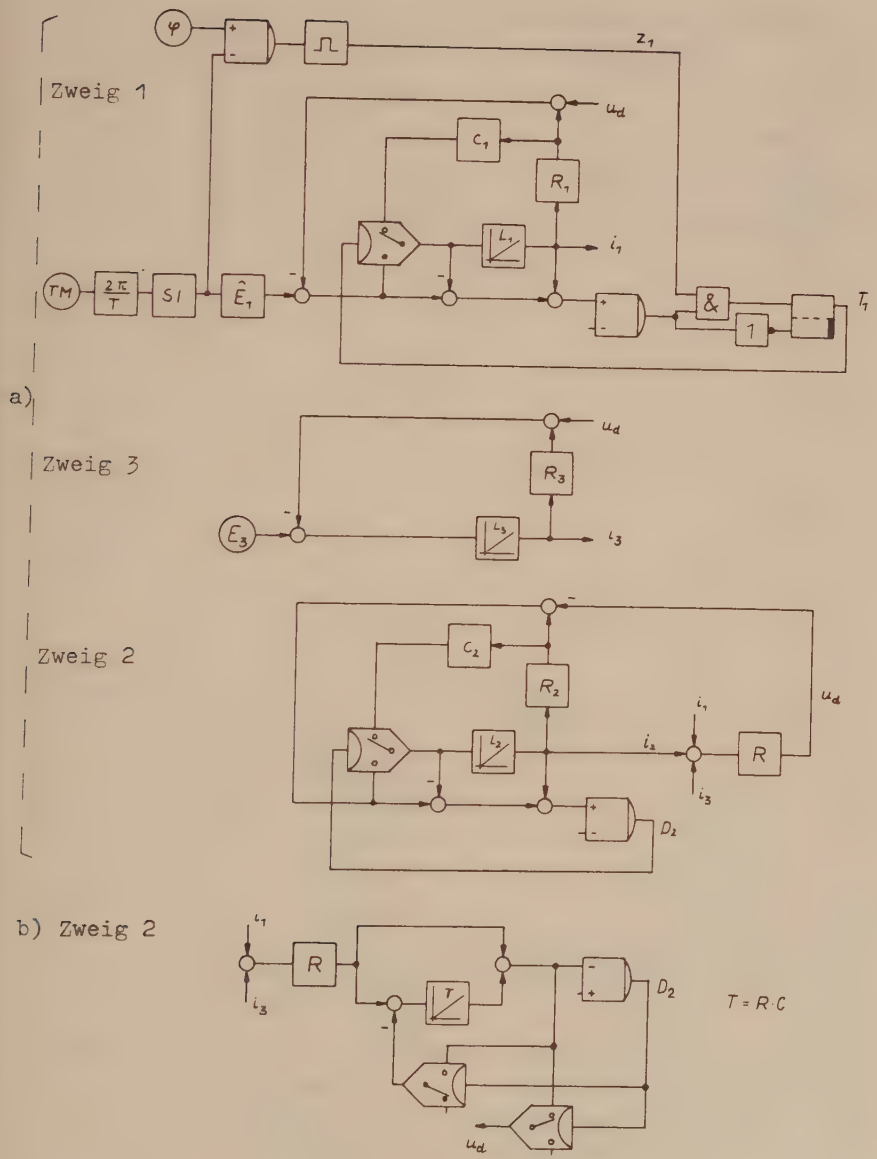


Bild 5.3.1/2 Blockschaltbild der Einpulsschaltung
a) mit Innenwiderstand im Zweig 2
b) Zweig 2 ohne Innenwiderstand

```
    EINPULSSCHALTUNG MIT FREILAUF VERS. A)
**STRUKTUR
UG11  TM                    NETZGROESSEN
SI11  UG11
UG02  SI11
KUG4  SIG1,EKG4
MFG1  KUG4
AD11  UK1G,UGG4                 ZWEIG 1
SU1G  AD11,UGG2
SW11  FFG1,UG11,SU1G
U1G1  SW11
SL11  SW11,SU1G
UG1G  U1G1
UG11  U1G1
AD1G  SL11,U1G1
KUG1  AD1G
UNG1  KUG1,MFG1
NIG1  KUL1
FFG1  UNG1,NIG1
SU21  UKG4,UG2G                 ZWEIG 2
SW12  KUG2,UG21,SU21
SU2G  SW12,SU21
U1G2  SW12
AD2G  SU2G,U1G2
UG21  U1G2
UG2G  U1G2
KUG2  AD2G
AD3G  UG3G,UGG4                 ZWEIG 3
SU3G  AD3G,EKG3
U1G3  SU3G
UG3G  U1G3
AD04  U1G1,U1G2,U1G3            REGELZWEIG
UGG4  AD04
EN
**PARAMETER                     **ANFANGSWERTE
UG11      0.0003142             EKG3 10.0
UG02      141.4                 EKG4 0.985
UKG4      10.0                  U1G1 -49.0
UG1G      +0.05                 U1G2 49.0
UG11      -15.0                 EN
UG3       +0.1                  **BEARBEITUNG
UL2G      -0.05                 NS
UG21      -15.0                 T1 5.0
U1G1Z     +1.0                  SD 3GGG.0
U1G1N               318.0       KU  5.0    U1G1,U1G3,U1G2,UGG4,UG12
U1G2Z     +1.0                  EN
U1G2N                31.8       FINIS
U1G3Z     +1.0
U1G3N               3180.0
MFG1       50.0
EN
```

Bild 5.3.1/3 DIGSI-Programm des Einpulsstromrichters mit Innenwiderstand im Zweig 2

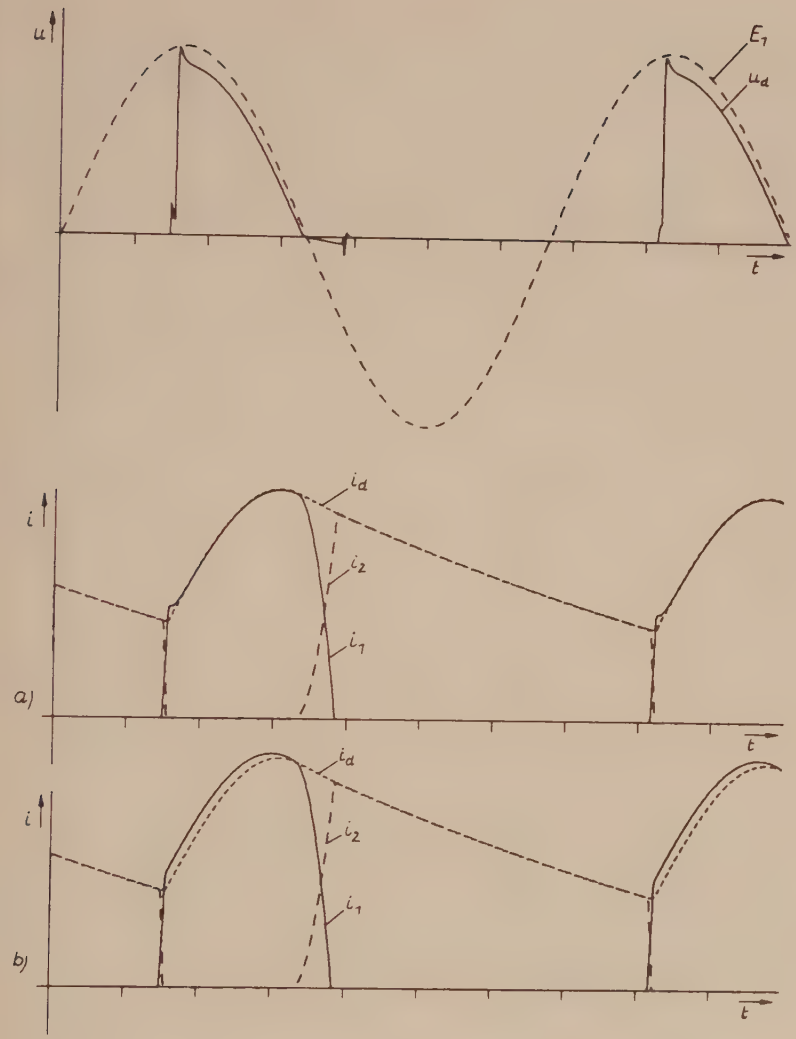


Bild 5.3.1/4 Einpulsschaltung mit Innenwiderstand im Zweig 2
Spannungs- und Stromverläufe
4000 Schritte pro Periode
a) PI-Regelung von u_d
b) P-Regelung von u_d

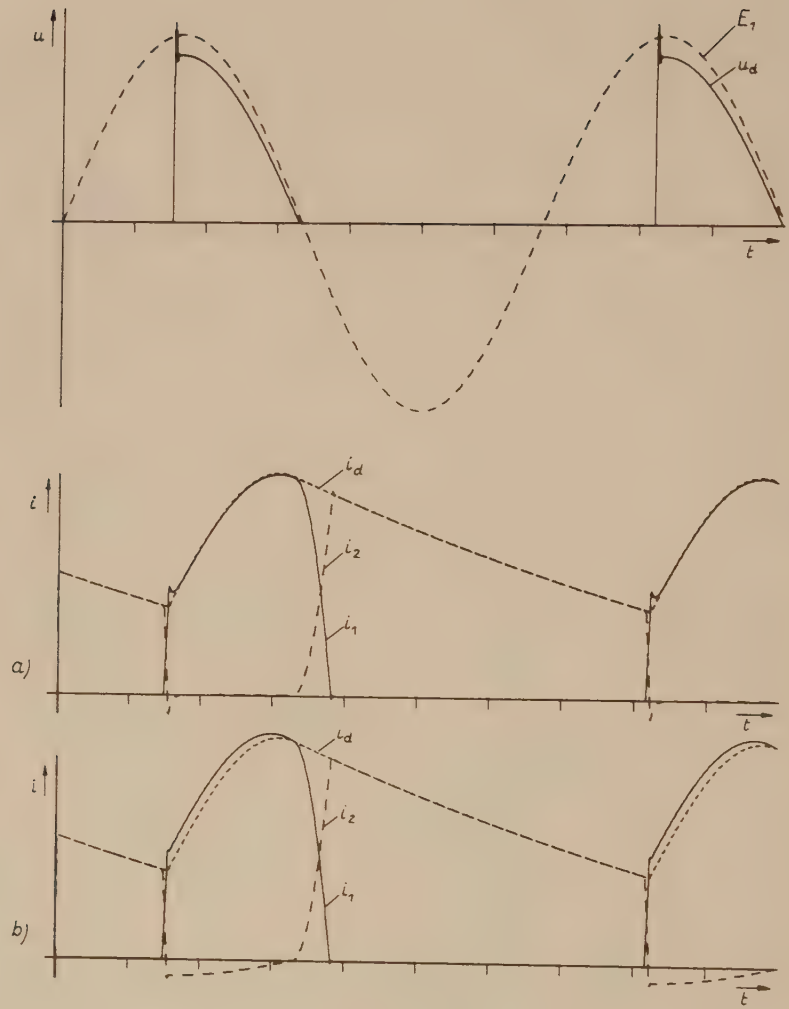


Bild 5.3.1/5 Einpulsschaltung ohne Innenwiderstand im Zweig 2
Spannungs- und Stromverläufe
1000 Schritte pro Periode
a) PI-Regelung von u_d
b) P-Regelung von u_d

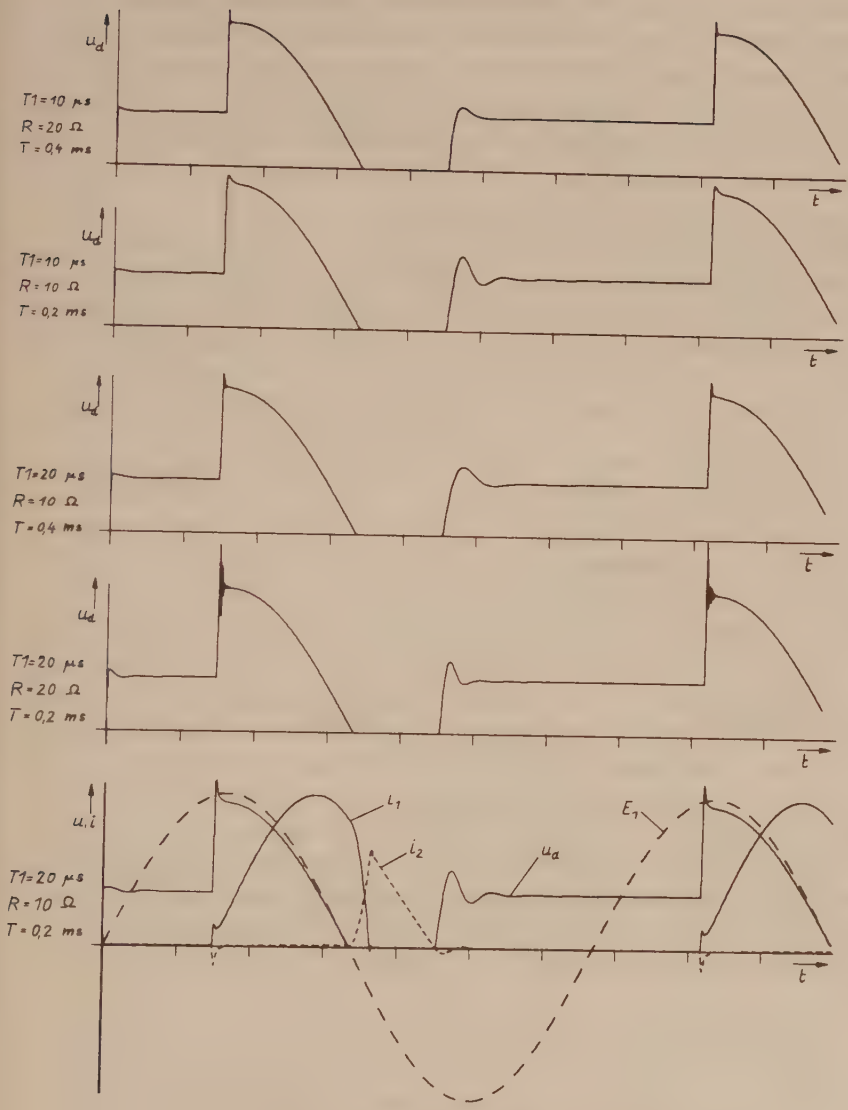
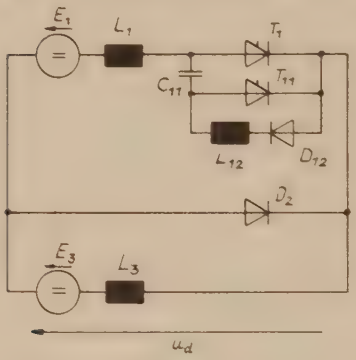


Bild 5.3.1/6 Einpulsschaltung ohne Innenwiderstand im Zweig 2
Verlauf der Gleichspannung u_d bei Variation der Simulationsparameter R, T und T1

Das Beispiel zeigt, daß die Simulation ohne Zusatzzweig einfacher und im Ergebnis günstiger ist. Die Einstellung der Schrittweite ist unkritisch. Ein integralwirkender Regler ist erforderlich, um den Simulationsfehler klein zu halten. Dabei können die Regelparameter angenähert abgeschätzt werden, um noch eine stabile Lösung zu erhalten.

5.3.2 Der Gleichstromsteller.

Anhand eines einfachen Gleichstromstellers in Bild 5.3.2/1 wird der Fall paralleler Ventilzweige einer Kommutierungsgruppe dargestellt. Die Schaltung entspricht einer Mittelpunktschaltung, bei der dem Thyristor T_1 die Löscheinrichtung mit den Ventilzweigen T_{11} und D_{12} parallelgeschaltet ist.



$V_1 = -E_1 + u_d \qquad L_1 \cdot \frac{d}{dt}(i_1) = -V_1 - u_{T1}$

$V_2 = u_d \qquad i_2 = \sum_{k=1}^{3} i_k - i_1 - i_3$

$V_3 = -E_3 + u_d \qquad L_3 \cdot \frac{d}{dt}(i_3) = -V_3$

Bild 5.3.2/1 Gleichstromsteller.

Da der Zweigstrom i_1 überlappungsfrei zwischen T_1 und T_{11} wechseln kann, lassen sich die Strom- und Spannungsverhältnisse für die parallelen Ventilzweige in Abhängigkeit von der Thyristorspannung des Ventils T_1 angeben.

$$u_{T11} = u_{T1} - u_{C11} \qquad u_{C11} = \frac{1}{C_{11}} \cdot \int (i_1 \cdot T_{11} - i_{12})\, dt \qquad (5.3.2/1)$$

$$u_{D12} = -u_{T11} - u_{L12} \qquad u_{L12} = -u_{T11} \cdot D_{12} = L_{12} \cdot \frac{d}{dt}(i_{12}) \qquad (5.3.2/2)$$

Anmerkung: im folgenden werden die Schaltfunktionen der Ventile mit T bzw. D bezeichnet.

Die übrigen Zweige werden wieder durch die allgemeinen Gleichungen für Mittelpunktschaltungen beschrieben. Dabei wurde der Freilaufzweig mit dem Hilfszweig zusammengefaßt.

Aus den Modellgleichungen ergibt sich das Blockschaltbild nach Bild 5.3.2/2. Auffällig ist der Mehraufwand an logischen Funktionen, um die Ventile T_1 und T_{11} gegenseitig für die Stromführung zu verriegeln. Aufgrund der unterschiedlichen Sperrspannungen von T_1 und T_{11} ist die Vorzeichenbestimmung des Ventilstromes von der Ventilspannung getrennt. Für stabilen Betriebszustand ist eine einfache Zweipunktregelung des Laststromes eingesetzt, die abwechselnd die Ventile T_1 und T_{11} zündet.

In Bild 5.3.2/3 ist der Löschvorgang des Ventils T_1 aufgezeichnet. Dazu sind angegeben der Zweigstrom i_1, der Strom im Umschwingkreis i_{12}, die Kondensatorspannung u_{C11} sowie die Sperrspannungen der Ventile. Der Zweigstrom i_1 wird bei $t=t_1$ unterbrechungsfrei auf den Thyristor T_{11} geschaltet, wodurch der Kondensator umgeladen wird; der Zweigstrom kommutiert bei $t=t_2$ auf den Freilauf. Die anschließende Teilentladung des Kondensators und Umladung mit Zünden des Ventils T_1 ergibt sich aus dem Verlauf des Umladestroms i_{12}. Zu bemerken ist, daß dieser Strom nicht in dem Zweigstrom i_1 oder in dem Laststrom i_d auftritt; der Umschwingkreis ist unabhängig von der Last- oder Netzseite. Der Kondensator wird aufgrund der grossen Entladung während des Freilaufs auf eine geringere negative Spannung aufgeladen, als sie vor dem Löschvorgang bestand. Der Betriebszustand ist insofern nicht stationär.

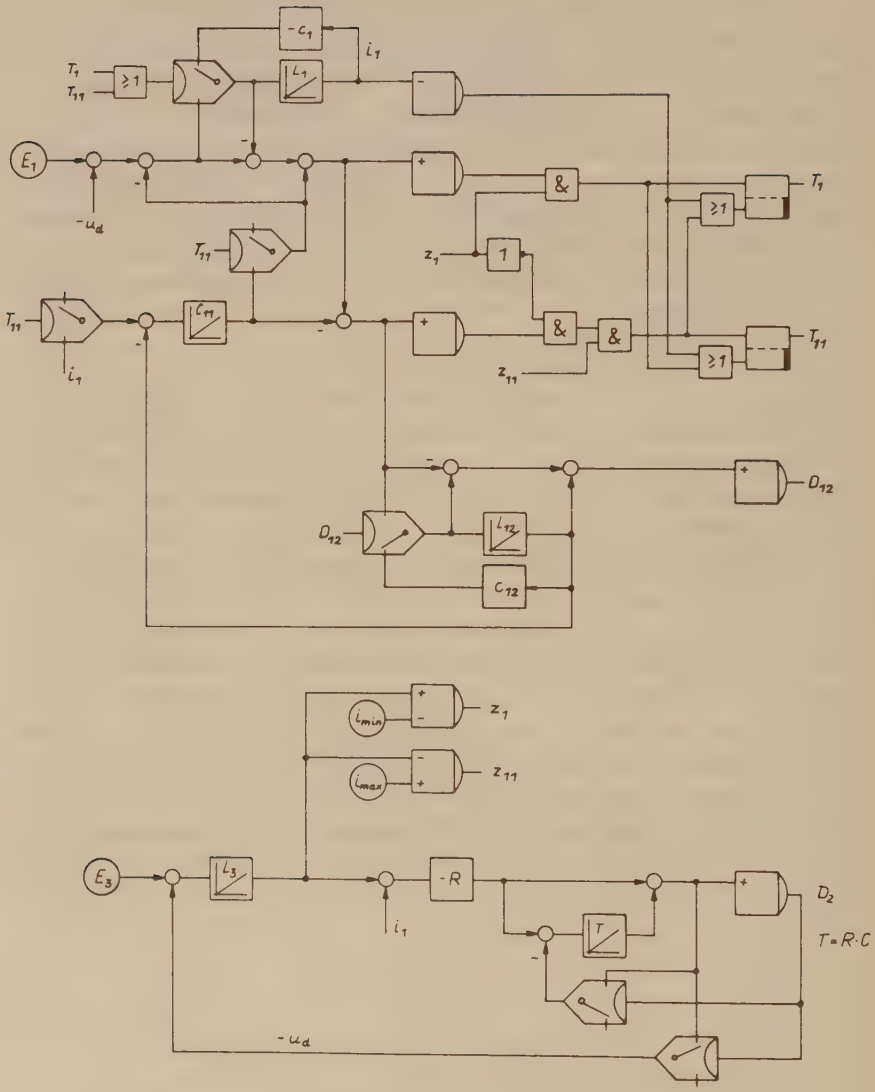


Bild 5.3.2/2 Blockschaltbild des Gleichstromstellers

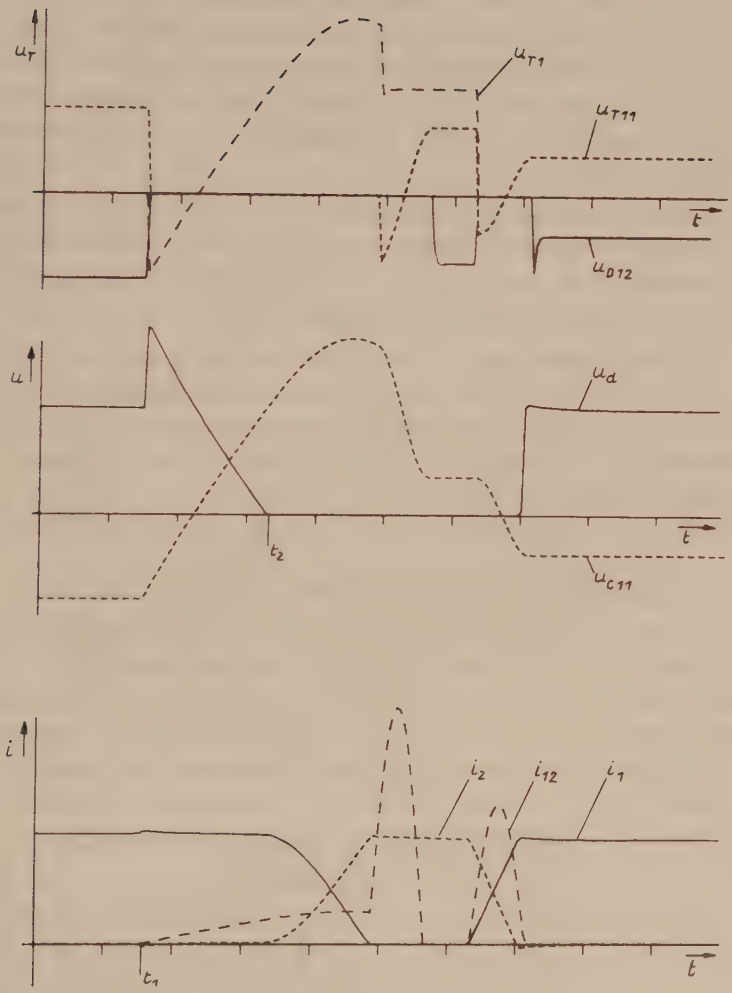


Bild 5.3.2/3 Gleichstromsteller
Spannungs- und Stromverläufe beim Löschen von Thyristor T_1

5.4 Die Simulation von Einphasenbrückenschaltungen.

Für den Antrieb von Bahnfahrzeugen werden in Verbindung mit Mischstrommotoren Stromrichter in Einphasenbrückenschaltung verwendet. Zur Herabsetzung der Grundschwingungsblindleistung wird der Stromrichter halbgesteuert ausgeführt. Üblicherweise werden heute zwei Stromrichter in Folgeschaltung eingesetzt, wodurch man eine Verbesserung des Blindleistungsverhaltens erreicht.

Eine weitere Senkung der Grundschwingungsblindleistung ist dann zu erwarten, wenn zwangskommutierte Stromrichter eingesetzt werden. Dazu wurde in /11/ die löschbare, unsymmetrische Brückenschaltung (LUB) vorgeschlagen.

Im folgenden werden eine halbgesteuerte und eine löschbare Einphasenbrückenschaltung simuliert. Als Beispiel für eine Auswertung der Ergebnisse werden Aussteuerung und Verschiebungsfaktor in Abhängigkeit von den Steuerwinkeln angegeben. Die Schaltungsdaten wurden dafür der Ausführung für den Triebzug ET 420 entnommen und sind im Anhang A3 aufgelistet.

5.4.1 Die unsymmetrisch halbgesteuerte Brückenschaltung.

Für Antriebe im motorischen Bereich am Einphasennetz eignet sich diese Schaltung besonders wegen des geringen Blindleistungsbedarfs infolge des über das Diodenzweigpaar möglichen Freilaufbetriebes und weil bei der unsymmetrischen Steuerung keine Kippgefahr im Bereich kleiner Aussteuerung besteht. Der Stromrichter in Bild 5.4.1/1a arbeitet auf einem Lastkreis mit der Motorgegenspannung E_d und wird von dem Einphasennetz E_a mit der Impedanz Z_a gespeist.

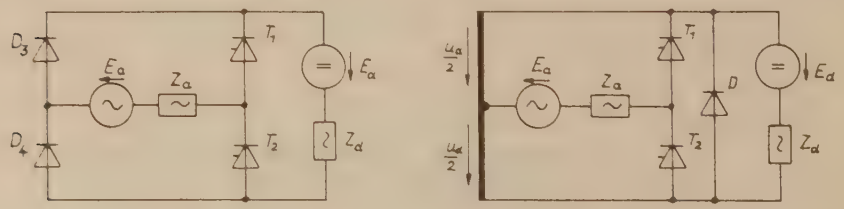


Bild 5.4.1/1 Vereinfachung der Einphasenbrückenschaltung.

Im Hinblick auf die Simulation läßt sich die Schaltung vereinfachen. Wenn man den Betrieb in der Nähe des Kurzschlusses ausschließt, bei dem alle vier Ventile leitend sein können, so sind an der Stromführung höchstens eines der steuerbaren Ventile T_1 oder T_2 und der Freilaufzweig (D_3 oder D_4) beteiligt. Die Funktion der Schaltung ändert sich nicht, wenn im Modell der Freilauf nur durch eine Diode ermöglicht wird, und der Richtbetrieb über den Thyristor und einen idealen Transformator als Ersatz für das Diodenzweigpaar. Die Schaltung wird damit auf die in Bild 5.4.1/1b dargestellte Anordnung übergeführt. Die Zahl der im Modell nachzubildenden Ventile ist um eins verringert.

Anhand des Bildes 5.4.1/2 lassen sich die Modellgleichungen ableiten.

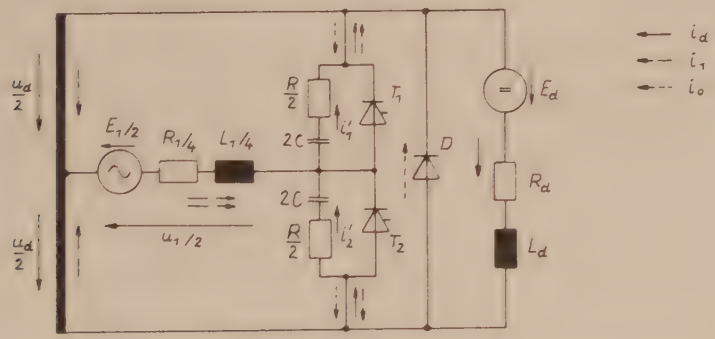


Bild 5.4.1/2 Schaltung zur Simulation der Einphasenbrücke.

Zur Systembeschreibung werden die Maschenströme i_1, i_d und i_o vorgegeben und den Thyristoren Hilfszweige als R-C-Reihenglied parallelgeschaltet. Um gleiche Leistungsverhältnisse wie bei der ursprünglichen Brücke zu erhalten, wird im Netzzweig mit halbierter Spannung und verdoppeltem Strom gerechnet.Es ergeben sich die folgenden Gleichungen.

Last: $u_d = -u_{T1} - u_{T2}$ $\qquad L_d \cdot \frac{d}{dt}(i_d) = -E_d + u_d - R_d \cdot i_d$ (5.4.1/1)

Netz: $u_1 = u_{T1} - u_{T2}$ $\qquad L_1 \cdot \frac{d}{dt}(i_1) = E_1 - u_1 - R_1 \cdot i_1$ (5.4.1/2)

Ventile:
$$u_{T1} = \frac{1}{2}\left(R \cdot i_1' + \frac{1}{C}\int i_1' dt\right) \qquad i_{T1} = (i_1 + i_d - i_o)\{T_1\}$$
$$u_{T2} = \frac{1}{2}\left(R \cdot i_2' + \frac{1}{C}\int i_2' dt\right) \qquad i_{T2} = (-i_1 + i_d - i_o)\{T_2\} \quad (5.4.1/3)$$
$$u_o = -u_d \qquad i_o = i_d - (i_d - i_o)$$

Richtbetrieb:

$$\{T_1\} = \begin{cases} 0 & i_1' = i_1 + i_d - i_o \\ 1 & i_1' = -\frac{1}{R}\, u_{C1} \end{cases} \qquad i_1' = (i_1 + i_d - i_o)\{\overline{T_1}\} - \frac{1}{R}\, u_{C1}\{T_1\}$$

(5.4.1/4)

$$\{T_2\} = \begin{cases} 0 & i_2' = -i_1 + i_d - i_o \\ 1 & i_2' = -\frac{1}{R} \cdot u_{C2} \end{cases} \qquad i_2' = (-i_1 + i_d - i_o)\{\overline{T_2}\} - \frac{1}{R} \cdot u_{C2}\{T_2\}$$

Kommutierung:

$$\{\overline{T_1 \vee T_2} \wedge D\} = 1 \qquad \begin{matrix} i_1' + i_2' = \frac{1}{R}(u_{C1} + u_{C2}) \\ i_1' + i_2' = 2 \cdot (i_d - i_o) \end{matrix} \qquad i_d - i_o = -\frac{1}{2R}(u_{C1} + u_{C2})$$

$$\{T_1 \wedge D\} = 1 \qquad i_d - i_o = i_1 - \frac{1}{R} \cdot u_{C2}$$

$$\{T_2 \wedge D\} = 1 \qquad i_d - i_o = -i_1 - \frac{1}{R} \cdot u_{C1}$$

$$i_d - i_o = -\frac{1}{2R}(u_{C1} + u_{C2})\{D \wedge \overline{T_1 \vee T_2}\} + (i_1 - \frac{1}{R} \cdot u_{C2})\{T_1 \wedge D\} - (i_1 + \frac{1}{R} \cdot u_{C1})\{T_2 \wedge D\} + i_d\{D\} \quad (5.4.1/5)$$

Anmerkung: die logischen Funktionen sind in geschweifte Klammern gesetzt und ergeben das arithmetische Äquivalent 1 oder 0.

Anhand der aufgestellten Gleichungen läßt sich das Blockschaltbild in Bild 5.4.1/3 entwickeln, das zur Beschreibung des Simulationsproblems in DIGSI herangezogen wird.

Die Zeitverläufe von Strömen und Spannungen der Netz- und Lastseite zeigt Bild 5.4.1/4 für einen stationären Betriebspunkt mit einem Anschnittwinkel von $\alpha = 60^{\circ}$. Die Schaltspitzen in der Gleichspannung u_d werden durch die R-C-Glieder verursacht, da beim Zünden des Ventils T_1 bzw. T_2 der entsprechende Kondensator aufgrund von L_1 auf eine höhere Spannung aufgeladen wird als die Netzspannung E_1.

Zur Bestimmung der Steuerkennlinie $a = E_d/\bar{u}_{di} = f(\alpha)$ wurde ein Regelkreis eingesetzt, der die Gegenspannung E_d bei konstantem Anschnittwinkel α so einstellt, daß der Nenngleichstrom fließt. Dabei läßt sich vorteilhaft die Parameterversion von DIGSI einsetzen.

Das Diagramm in Bild 5.4.1/5 zeigt die Aussteuerung a und die Verschiebung der Grundschwingung des Netzstromes $\cos\varphi_1$ in Abhängigkeit vom Steuerwinkel α bei $I_d = I_{dN}$. Dem gegenübergestellt sind die Funktionen unter idealen Verhältnissen.

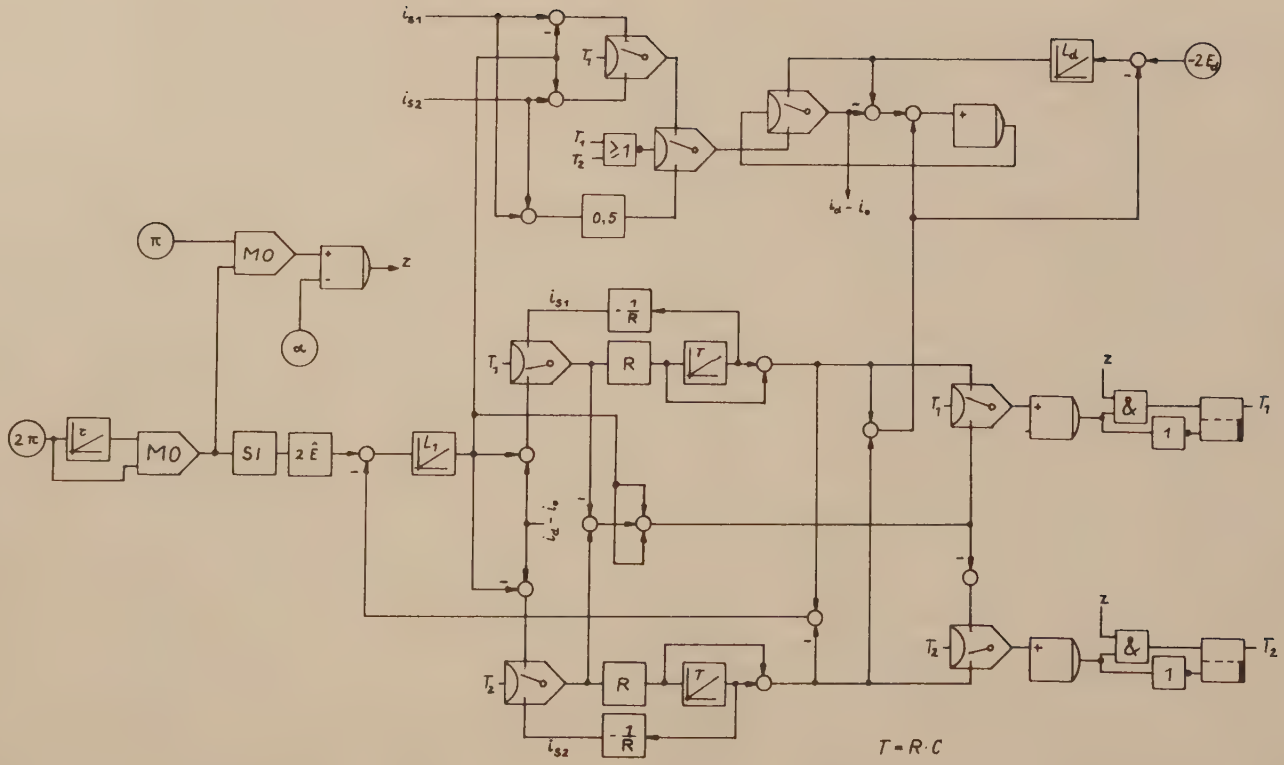


Bild 5.4.1/3 Blockschaltbild der Einphasenbrücke

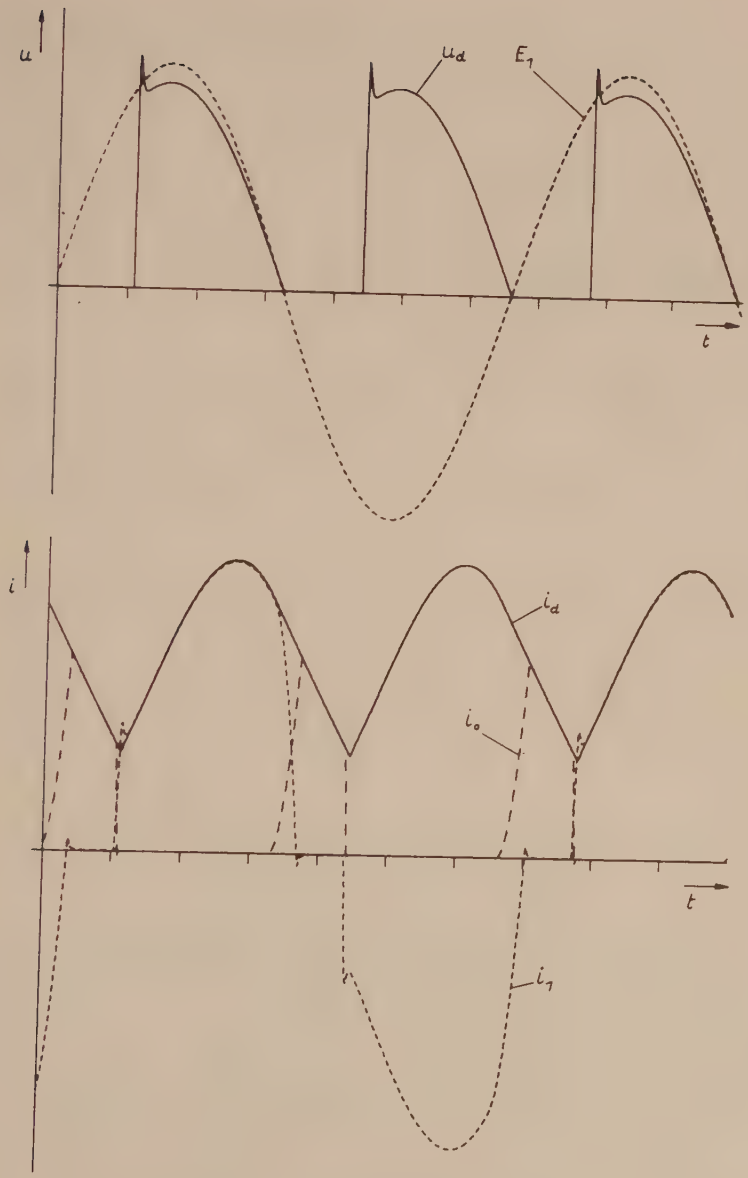


Bild 5.4.1/4 Halbgesteuerte Einphasenbrücke
Spannungs- und Stromverläufe bei $\alpha = 60^{\circ}$

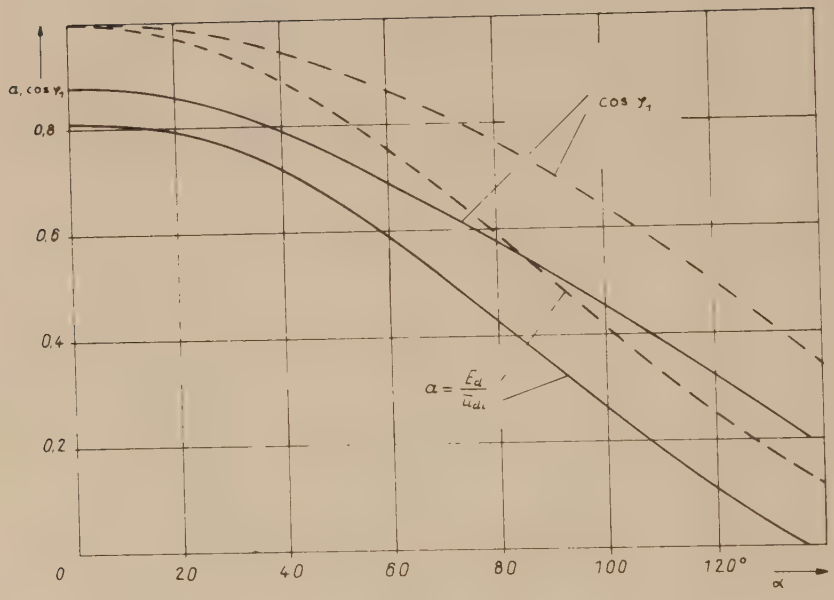


Bild 5.4.1/5 Halbgesteuerte Einphasenbrücke
Aussteuerung und Phasenverschiebung der Grundschwingung des Netzstromes in Abhängigkeit vom Zündwinkel α
$I_d = I_{dN}$, $\cos\varphi_1$ induktiv
--- Idealfall, $L_d=\infty$, $R_d=R_1=L_1=0$
— mit den Daten des Triebzuges ET 420

5.4.2 Die löschbare, unsymmetrische Brückenschaltung.

Um bei einer halbgesteuerten Brückenschaltung den Leistungsfaktor zu erhöhen, wird in /11/ eine ergänzende Einrichtung angegeben, die innerhalb einer Halbschwingung der Netzspannung den Thyristorstrom zum Verlöschen bringen kann. Die Schaltung nach Bild 5.4.2/1a enthält neben der eigentlichen Löscheinrichtung mit dem Kondensator C_L einen Speicherkondensator C_S, um die Energie beim Verlöschen des Netzstromes aufzunehmen.

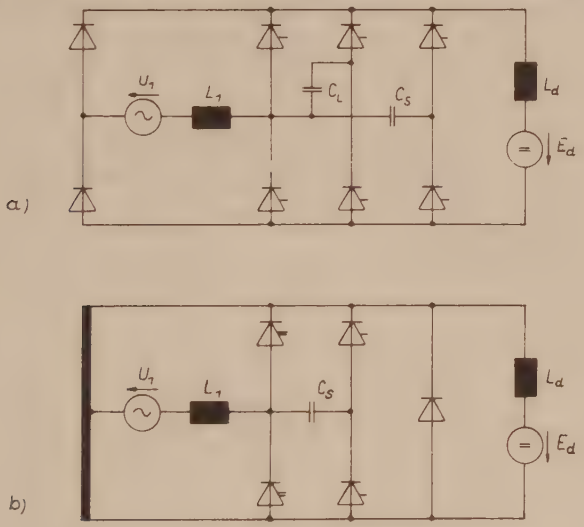


Bild 5.4.2/1 Die löschbare Einphasenbrücke
a) Schaltung
b) Modellschaltbild

Bei den mit Bahnfrequenz betriebenen Stromrichtern löschen die selbstgeführten Ventile in Zeiten, die nur ein kleines Vielfaches der Freiwerdezeiten betragen; die Netz- und Gleichgrößen werden nur unwesentlich davon beeinflußt. Unter dieser Vorraussetzung kann die Löscheinrichtung für die Simulation durch abschaltbare Thyristoren ersetzt werden. Man erhält dann die Schaltung nach Bild 5.4.2/1b, in der die Funktion des Speicherkondensators erhalten geblieben ist. Zur Ableitung der Modellgleichungen sind wie im Abschnitt 5.4.1 zwei Hilfszweige eingefügt und die Maschenströme i_1, i_d und i_o vorgegeben.

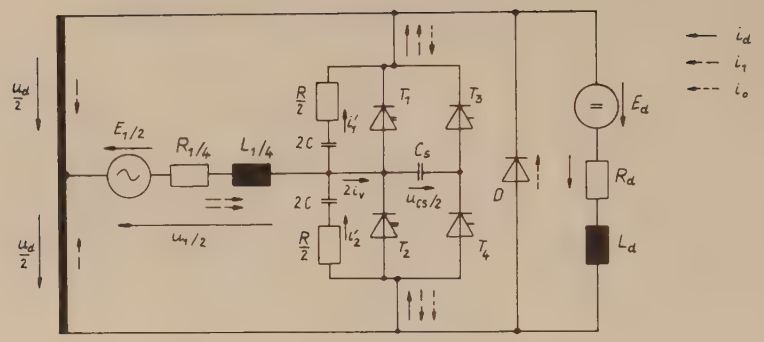


Bild 5.4.2/2 Schaltbild der löschbaren Brücke zur Ableitung der Modellgleichungen

Anhand des Schaltbildes (Bild 5.4.2/2) ergeben sich die folgenden Gleichungen.

Netz- und Lastkreis:

$$u_1 = u_{T1} - u_{T2} \qquad L_1 \cdot \frac{di_1}{dt} = E_1 - u_1 + R_1 \cdot i_1 \qquad (5.4.2/1)$$

$$u_d = -u_{T1} - u_{T2} \qquad L_d \cdot \frac{di_d}{dt} = -E_d + u_d - R \cdot i_d \qquad (5.4.2/2)$$

Spannungen und Ströme der Ventile:

$$\begin{aligned}
u_{T1} &= \tfrac{1}{2}\left(R \cdot i_1' + \tfrac{1}{C}\int i_1' dt\right) & i_{T1} &= (i_1 + i_d - i_o)\{T_1\} \\
u_{T2} &= \tfrac{1}{2}\left(R \cdot i_2' + \tfrac{1}{C}\int i_2' dt\right) & i_{T2} &= (-i_1 + i_d - i_o)\{T_2\} \\
u_{T3} &= u_{T1} - \tfrac{1}{2} u_{cs} & i_{T3} &= (i_1 + i_d - i_o)\{T_3\} \\
u_{T4} &= u_{T2} + \tfrac{1}{2} u_{cs} & i_{T4} &= (-i_1 + i_d - i_o)\{T_4\} \\
u_o &= -u_d & i_o &= i_d - (i_d - i_o)
\end{aligned} \qquad (5.4.2/3)$$

Kondensator:

$$\begin{aligned}
u_{cs} &= \frac{1}{2C}\int 2 i_{cs}\, dt & 2 i_{cs} &= (2 i_1 - i_1' + i_2')\{T_3 \vee T_4\} \\
u_{cs}^* &= u_{cs}\{T_3 \vee T_4\}
\end{aligned} \qquad (5.4.2/4)$$

Richtbetrieb:

$$\{T_1 \vee T_3\} = \begin{cases} 0 & i_1' = i_1 + i_d - i_o \\ 1 & i_1' = \frac{1}{R}(-u_{C1} + u_{CS}\{T_3 \vee T_4\}) \end{cases} \qquad i_1' = (i_1 + i_d - i_o)\{\overline{T_1 \vee T_3}\} - \frac{1}{R}(u_{C1} - u_{CS}^*)\cdot\{T_1 \vee T_3\}$$

(5.4.2/5)

$$\{T_2 \vee T_4\} = \begin{cases} 0 & i_2' = -i_1 + i_d - i_o \\ 1 & i_2' = \frac{1}{R}(-u_{C2} - u_{CS}\{T_3 \vee T_4\}) \end{cases} \qquad i_2' = (-i_1 + i_d - i_o)\{\overline{T_2 \vee T_4}\} - \frac{1}{R}(u_{C2} + u_{CS}^*)\{T_2 \vee T_4\}$$

Kommutierung:

$\{\overline{T_1 \vee T_2 \vee T_3 \vee T_4} \wedge D\} = 1$ $\qquad i_1' + i_2' = -\frac{2}{R}(u_{C1} + u_{C2})$

$$i_d - i_o = -\frac{1}{R}(u_{C1} + u_{C2})$$

$$i_1' + i_2' = 2(i_d - i_o)$$

$\{(T_1 \vee T_3) \wedge D\} = 1$ $\qquad i_1' \cdot R + u_{C1} - u_{CS}^* = 0$, $\quad i_1' - i_1 - i_d + i_o = 0$ $\qquad i_d - i_o = -i_1 - \frac{1}{R}(u_{C1} - u_{CS}^*)$

$\{(T_2 \vee T_4) \wedge D\} = 1$ $\qquad i_2' R + u_{C2} + u_{CS}^* = 0$, $\quad i_2' + i_1 - i_d + i_o = 0$ $\qquad i_d - i_o = i_1 - \frac{1}{R}(u_{C2} + u_{CS}^*)$

$$i_d - i_o = \left[-\frac{1}{R}(u_{C1} + u_{C2})\{\overline{T_1 \vee T_2 \vee T_3 \vee T_4}\} + (i_1 - \frac{1}{R}(u_{C2} + u_{CS}^*))\{T_2 \vee T_4\} - (i_1 + \frac{1}{R}(u_{C1} - u_{CS}^*))\{T_1 \vee T_3\}\right]\{D\} + i_d\{\bar{D}\}$$

(5.4.2/6)

Anmerkung: die logischen Funktionen sind in geschweifte Klammern gesetzt und ergeben das arithmetische Äquivalent 1 oder 0.

Aus den Gleichungen (5.4.2/1) bis (5.4.2/6) kann das Blockschaltbild entwickelt werden, das in Bild 5.4.2/3 dargestellt ist. Im Vergleich zur einfachen anschnittgesteuerten Einphasenbrücke erweitert sich das Blockschaltbild um die Darstellung zweier löschbarer Ventile und des Speicherkondensators sowie die damit zusammenhängenden Verknüpfungen der Gleichungen. Die Simulation erfordert somit keinen wesentlichen Mehraufwand.

Für einen stationären Betriebspunkt sind die wichtigen Zeitverläufe von Spannungen und Strömen der Schaltung in Bild 5.4.2/4 aufgetragen. Anhand der aufgezeichneten Ventilspannungen von T_1 und T_3 läßt sich die Funktion der Schaltung verfolgen; u_{T2} und u_{T4} haben den gleichen Zeitverlauf um eine halbe Netzperiode versetzt.

Da die Simulation nur die Wirkung des Speicherkondensators zeigen soll, ist, wie bereits erwähnt, der eigentliche Löschvorgang der Ventile nicht enthalten. Die Stromführung wechselt daher kommutierungsfrei vom Hauptthyristor auf den Löschthyristor und umgekehrt. Mit dem Löschwinkel β übernimmt der Löschthyristor den Netzstrom, und der Speicherkondensator wird aufgeladen. Überschreitet u_{CS} die Netzspannung, geht der Laststrom auf den Freilauf über, dessen Kommutierung bei γ beendet ist. Aufgrund der höheren Kondensatorspannung kann nun bereits vor dem Nulldurchgang der Netzspannung der andere Löschthyristor bei α gezündet werden, wie es der Netzstromverlauf zeigt. Wechselt die Netzspannung vor der Entladung des Kondensators ihr Vorzeichen, kann der Netzstrom auf den Hauptthyristor übergehen; im anderen Fall wechselt der Laststrom zunächst auf den Freilauf. Man erkennt, daß der Netzstrom kapazitiv phasenverschoben ist. Aussteuerung und Phasenverschiebung können über den Zündwinkel α und den Löschwinkel β verstellt werden.

In Abhängigkeit von dem Löschwinkel wurde für $I_d = I_{dN}$ die Aussteuerung $a = E_d/\bar{u}_{di}$ bestimmt. Dabei wurde der noch freie Parameter α einmal konstant gehalten ($\alpha = -25^o$) und zum anderen so eingestellt, daß der Speicherkondensator dabei so entladen wird, daß die Entladekurve mit dem Netznulldurchgang

zusammenfällt. Zu beiden Kurven wurde die Verschiebung der Grundschwingung des Netzstromes bestimmt.

Wie das Ergebnis in Bild 5.4.2/5 zeigt, ist die Phasenverschiebung in dem gesamten Aussteuerbereich kapazitiv. Bei veränderlichem Zündwinkel α nach dem genannten Steuerprinzip wird eine größere Phasenverschiebung erreicht, wobei der überstreichbare Aussteuerbereich erhalten bleibt.

Da die löschbare Brückenschaltung kapazitiv wirkt, kann mit maximal kapazitiver Steuerung die induktive Blindleistung einer einfachen Einphasenbrücke bei einer Folgeschaltung zum großen Teil kompensiert werden.

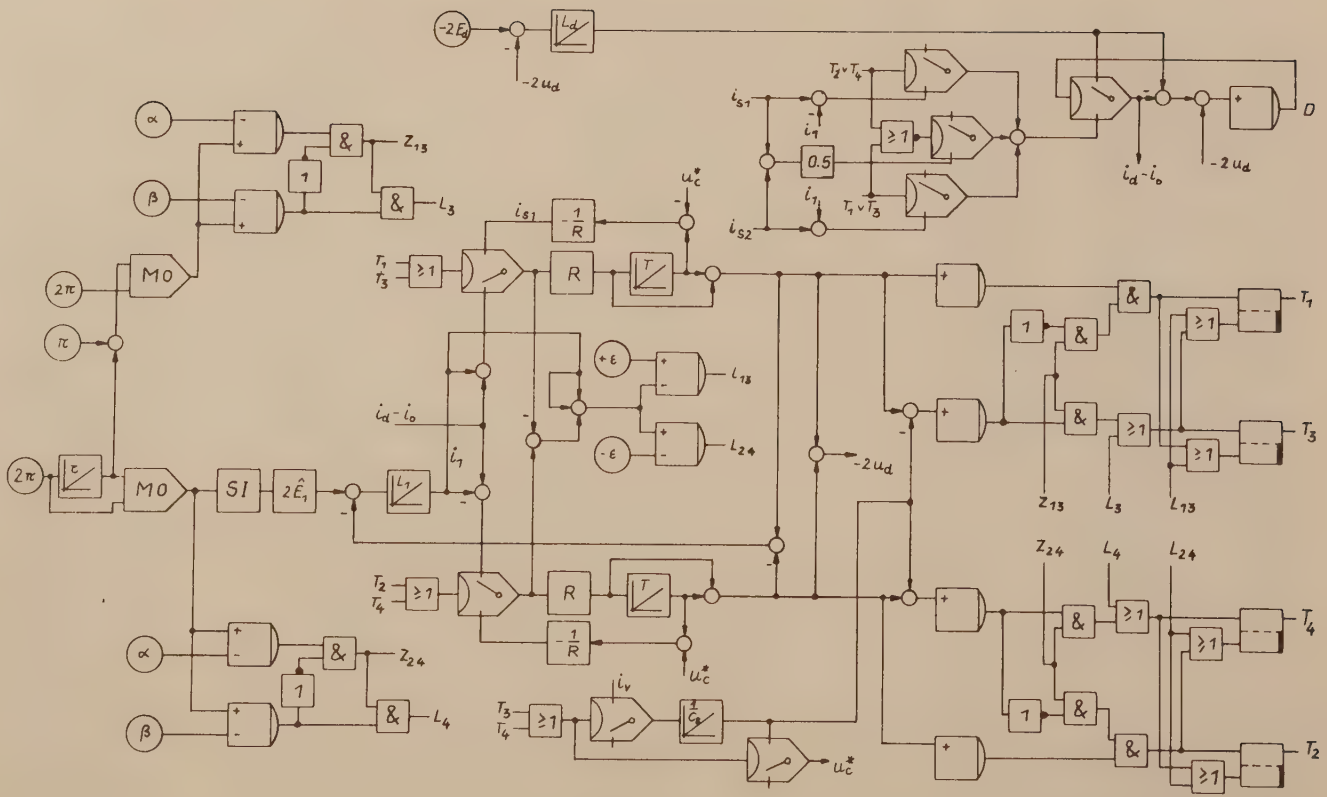


Bild 5.4.2/3 Blockschaltbild der löschbaren Einphasenbrücke

T · R·C

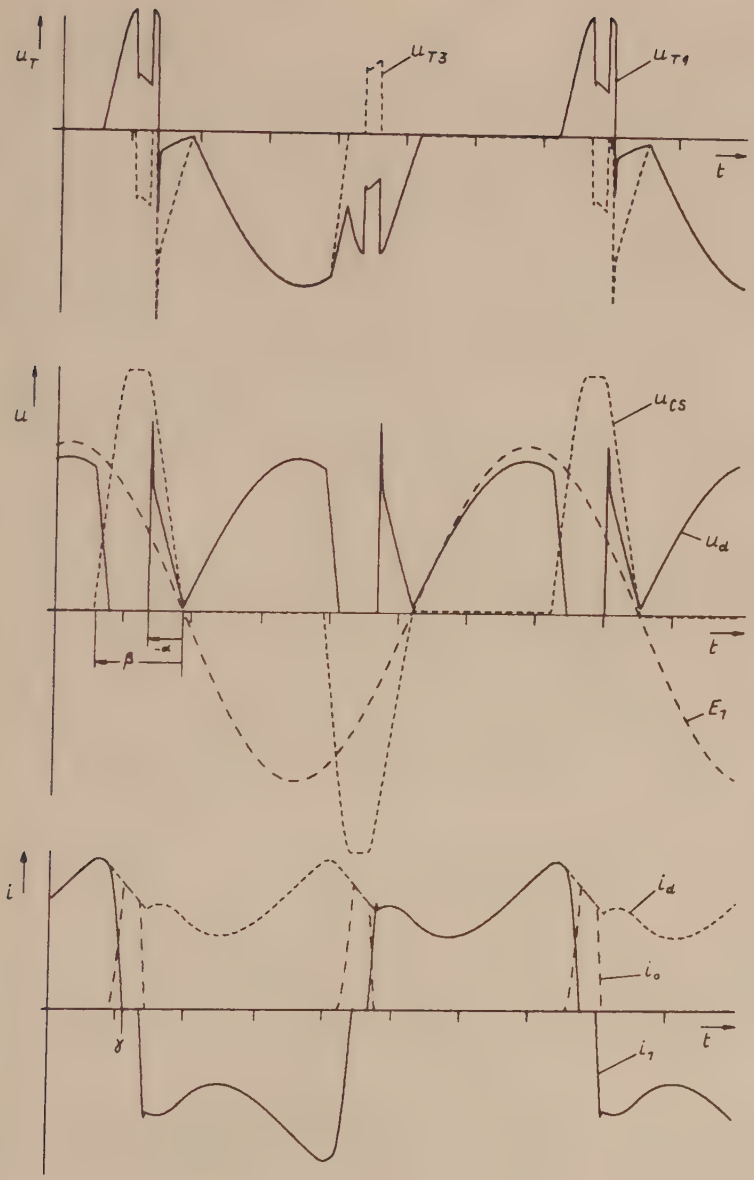


Bild 5.4.2/4 Löschbare Einphasenbrücke
Spannungs- und Stromverläufe
bei $\alpha = -34^{\circ}$ und $\beta = 70^{\circ}$

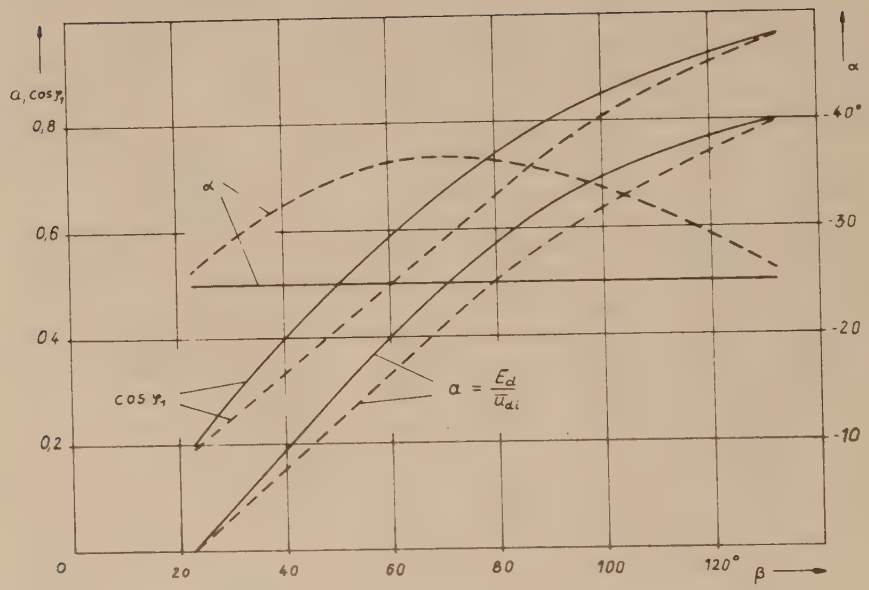


Bild 5.4.2/5 Löschbare Einphasenbrücke
Aussteuerung, Phasenverschiebung der Grundschwingung des Netzstromes und Zündwinkel α in Abhängigkeit vom Löschwinkel β
$I_d = I_{dN}$, $\cos\varphi_1$ kapazitiv
—— konstanter Zündwinkel α
---- Kondensator im Nulldurchgang der Netzspannung vollständig entladen

6 Zusammenfassung.

Stromrichterschaltungen sind mit allgemeinen Simulationsprogrammen sehr aufwendig darzustellen, da diese Programme in der Regel nur für kontinuierliche, zeitinvariante Systeme vorgesehen sind. Zwar gibt es spezielle Programme, mit deren Hilfe auch zeitvariante, elektrische Netzwerke darstellbar sind; sie sind jedoch hauptsächlich für die Simulation quasistationärer Vorgänge in den Netzwerken entwickelt worden und haben nicht die Allgemeingültigkeit der vorgenannten Programmarten. Für beide gilt, daß sie fast ausschließlich für Großrechenanlagen konzipiert sind.

Ausgehend von einem vorhandenen Simulationsprogramm (DIGSI) wurde ein Konzept für ein allgemein anwendbares Simulationssystem erarbeitet, das auch auf kleineren Rechenanlagen lauffähig ist. Da die Rechenzeit bei diesen Rechnern für den einzelnen Benutzer weniger beschränkt ist, kann eine längere Laufzeit des Programms in Kauf genommen werden. Das Simulationsprogramm wurde in zwei Versionen erstellt. In der Dialogversion kann das zu bearbeitende Problem entwickelt und getestet werden. Für die Auswertung von verschiedenen Betriebszuständen unter Änderung bestimmter Problemparameter kann über eine automatisch ablaufende Version verfügt werden. Durch Optimierung des Programmaufbaus und Abfassen des wesentlichen Rechenteils in der Maschinensprache konnte die Laufzeit auf 1/3 gegenüber der ursprünglichen Fassung gesenkt werden.

Für die Simulation einer bestimmten Klasse von Stromrichterschaltungen wurde ein Schema entwickelt, nach dem das erforderliche mathematische Modell für eine Simulation leicht aufgestellt werden kann. Dies wurde an einigen Beispielen gezeigt.

Für die löschbare, unsymmetrisch gesteuerte Einphasenbrückenschaltung wurde für eine noch nicht behandelte Steuerungsart die Steuerkennlinie bestimmt.

7 Schrifttum

/1/ Depenbrock, M. Skript zur Vorlesung Leistungselektronik Bochum, 1971

/2/ Möltgen, G. Netzgeführte Stromrichter mit Thyristoren Siemens, München, 1974 3.Aufl.

/3/ Kail, H.-J. Ein Beitrag zur Systematisierung der Simulation bekannter Stromrichterschaltungen vorwiegend mittels Analogrechners Dissertation, Bochum, 1974

/4/ Latzel, W. Wiegand, R. Digitale Simulation kontinuierlicher Prozesse. DIGSI-Handbuch für den Anwender. BBC, Mannheim

/5/ Latzel, W. Ein digitales Simulationsverfahren, dessen Koeffizienten direkt aus der Übertragungsfunktion des stetigen Systems berechnet werden können. Elektr. Datenverarb., 1970, S.406-409

/6/ Latzel, W. Vergleich von DIGSI mit anderen Simulationssprachen. BBC-Notiz AN 146

/7/ Jentsch, W. Digitale Simulation kontinuierlicher Systeme. Oldenbourg-V., München, 1969

/8/ Hamming, R. Numerical methods for scientists and engineers. Mc. Graw Hill, London, 1962

/9/ Depenbrock, M. Einphasenstromrichter mit optimiertem Leistungsfaktor. ETZ-A, 95(1974), S.360-363

/10/ Förster, J. Zur Stromrichternetzbelastung. ETZ-A, 96(1975), S.52-57

/11/ Förster, J. Sektorsteuerung mit löschbaren Stromrichterbrücken.
Techn. Rundschau 65(1973) S.25-29

/12/ Bezold, K.-H. Löschbare, einphasige Stromrichter mit Sektorsteuerung für die Speisung von Mischstrommotoren.
Elektr. Bahnen 45(1974) S.281-286

/13/ Skudelny, H. Analyse der halbgesteuerten Einphasenbrückenschaltung.
Arch. f. Elektr. 55(1972) S.44-56

/14/ Hoffmann, D. Ein Beitrag zur automatischen Simulation von Stromrichterschaltungen mittels Digitalrechner.
Dissertation, Berlin, 1974

/15/ Eisenack, H.
Hoffmeister, H. Digitale Nachbildung von elektrischen Netzwerken mit Dioden und Thyristoren.
Arch. f. Elektr. 55(1972) S.32-43

/16/ Albers, H. Beschreibung DIGSI 300
Bericht am Lehrstuhl für Erzeugung und Anwendung elektrischer Energie,
Bochum, 1976

A1 DIGSI-Syntax.

Zur Definition der Simulationssprache werden folgende Abkürzungen verwendet:
das Zeichen '::=' steht für 'wird definiert durch'.
Ausdrücke in runden Klammern geben eine Folge an, die mit einem Maximalwert versehen sein können.
Ausdrücke in spitzen Klammern geben den Bezeichner an.
Das Zeichen '/' steht als Trennungszeichen in einer Definitionsfolge.

```
Problembeschreibung::= (Anweisungsgruppe),Ende
Ende::= FI
Anweisungsgruppe::= Gruppenzeichen,(Anweisung),Endezeichen
Endezeichen::= EN
Anweisung::= Struktur/Parameter/Anfangswert/Bearbeitung
Gruppenzeichen::= ++ST/++PA/++AN/++BE
Struktur::= Funktion,Intervall,Funktion,b,(Funktion,b)=max. 2
Parameter::= Funktion,Identifikation,Wert,(Wert)=max. 3
Anfangswert::= Funktion,Wert
Bearbeitung::= Kennzeichen,Wert,(Funktion)=max. 5
Funktion::= Grundfunktion,Nummer
Grundfunktion::= AD/SU/UO/U1/U2/TZ/EK/WU/MO/SI/AB/IN/FG/MU/
                 DI/BG/SW/TS/KO/FF/MF/UN/OR/NI/NO/NA/TM/TA/IT
Nummer::= <Ziffer,Ziffer>
Wert::= <reelle Zahl>
Intervall::= ␣␣/+␣/++␣
Identifikation::= Z␣/N␣/<Ziffer>␣/
Kennzeichen::=T1/T2/T3/SD/AU/PU/PM/PO/KU/NS

b:= beliebiges Trennungszeichen
␣:= Leerzeichen
+:= Zeichen 'Stern'
```

Anhang A2

Die Bestimmung von integralen Meßwerten.

Bei der Ermittlung der Steuerkennlinien werden Meßgrößen benötigt, die sich aus dem integralen Wert über eine Periode ergeben. Da die Integratoren des Simulationssystems DIGSI nicht zurückgesetzt werden können, ohne daß die Simulation auf den Anfangszustand gesetzt wird, wird dazu folgendes allgemeine Verfahren abgeleitet.

Es sei $f(x) = f(x+T)$ eine periodische Funktion und der Integralwert über eine Periode $I = \frac{1}{T}\int_{(T)} f(x)\,dx$. Für die Berechnung von I wird nicht die Funktion f(x) integriert, sondern die Abweichung gegenüber dem vorhergehenden Integralwert. Für die Abweichung ergibt sich:

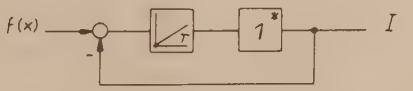
$$a = \frac{1}{T}\cdot\int_{nT}^{(n+1)\cdot T}\left[f(x) - I_{(nT)}\right]dx \qquad (A2/1)$$

$$a = \frac{1}{T}\cdot\int_{(T)} f(x)\,dx \; - \; \frac{1}{T}\cdot\int_{(T)} I_{(nT)}\,dx$$

$$a = I_{((n+1)T)} \; - \; I_{(nT)} \qquad (A2/2)$$

Nach jeder Periode hat der Integrator den richtigen Wert, der der unveränderte, alte Wert $I_{(nT)}$ ist, wenn sich a=0 ergibt. Zu dieser Messung gehört das einfache Blockschaltbild.

Dabei läßt sich die Abtastung in der Simulation mit DIGSI leicht durch einen Verstärker realisieren, der die Verstärkung V = 1 hat und mit der Schrittweite T_2 = T gerechnet wird.

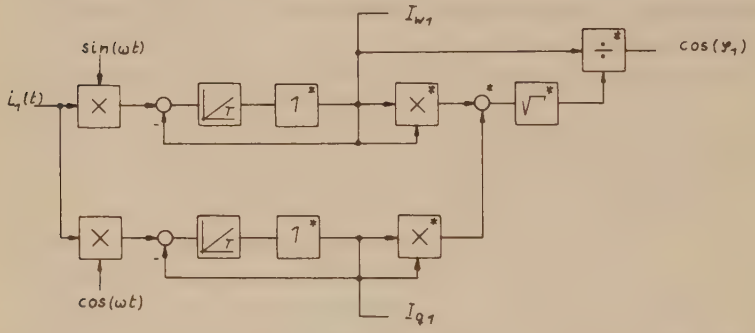


*... Schrittweite T_2

Aus dieser Grundschaltung ergeben sich die folgenden Meßschaltungen.

a) Mittelwert des Gleichstromes i_d.

b) Blind- und Wirkanteil des Netzstromes i_1 sowie der Phasenverschiebungsfaktor $\cos(\varphi_1)$.



Hierbei muß jedoch vorrausgesetzt werden, daß der Netzstrom i_1 zu wenigstens einem Zeitpunkt innerhalb einer Periode ungleich Null ist, da sonst die Phasenverschiebung nicht richtig berechnet werden kann.

A3 Daten des Triebzuges ET 420.

Die Speisung des Zuges erfolgt über einen Zweiwicklungstransformator auf je eine Einphasenbrücke, die in Folgeschaltung geschaltet sind. Die Stromrichter versorgen 6 Antriebsmotoren, die über je eine Glättungsdrossel parallelgeschaltet sind.

1. Transformator (je Wicklung):

U_1/U_2 = 15000V/448V	f = 16 2/3 Hz	
$I_{Nsek.}$ = 1220A	u_k = 7%	
Kurzschlußimpedanz	Z_k = 0,0259 Ω	
	X_k = 0,0243 Ω	L_k = 0,232 mH
	R_k = 0,009 Ω	

2. Glättungsdrossel:

L_g = 10,5 mH $\quad R_g$ = 0,04 Ω

3. Motor:

U_N = 600V; I_N = 335A

R_a = 0,06 Ω

L_a = 0,776 mH

4. Kondensatoren der LUB:

Speicherkondensator	C_S = 8,63 mF
Löschkondensator	C_L = 3,90 mF

Für die Simulation werden daraus folgende Daten übernommen:

Netzseite:

E_1 = 448V $\quad L_1$ = 0,232mH $\quad R_1$ = 0,01 Ω

Gleichspannungsseite:

$I_{dN} = 6 \cdot I_N$ = 2010A

$L_d = 1/6(L_g+L_a)$ = 1,883mH

$R_d = 1/6(R_g+R_a)$ = 0,0167Ω

Kondensator:

$C = C_S + 1/2C_L$ = 10,5mF

Lebenslauf

Hans Albers

geboren am	14. September 1940 in Marne
Eltern:	Arthur Albers, Elektromeister Margaretha Albers geb. Mall
Ausbildung:	
Volksschule	1947 - 1951 in Marne
Gymnasium	1951 - 1960 neusprachlich-naturwissenschaftliches Gymnasium in Marne
Praktikum	1962 Siemens AG in Hamburg 1965 Frieseke & Hoepfner in Erlangen 1967 Deutsche Forschungsanstalt für Luft- und Raumfahrt in Braunschweig
Studium	1962 - 1969 Technische Hochschule in Braunschweig Fachrichtung Elektrotechnik Diplom: April 1969
Wehrdienst:	1960 - 1961
Berufstätigkeit:	1969 - 1975 wiss. Assistent am Lehrstuhl für Erzeugung und Anwendung elektrischer Energie Ruhruniversität Bochum seit 1975 wiss. Angestellter am gleichen Lehrstuhl